
General Concepts for Human Supervision of Autonomous Robot Teams



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Vom Fachbereich Informatik der
Technischen Universität Darmstadt
zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte

Dissertation

von

Dipl.-Math. Karen Petersen
(geboren in Langen)

Referent: Prof. Dr. Oskar von Stryk
Koreferent: Prof. Dr. Daniele Nardi
Sapienza Università di Roma, Italien

Tag der Einreichung: 23. Mai 2013
Tag der mündlichen Prüfung: 09. Juli 2013

D17
Darmstadt 2014

Please cite this document as
URN: urn:nbn:de:tuda-tuprints-38735
URL: <http://tuprints.ulb.tu-darmstadt.de/3873>

This document is provided by tuprints,
E-Publishing-Service of the TU Darmstadt
<http://tuprints.ulb.tu-darmstadt.de>
tuprints@ulb.tu-darmstadt.de

Contents

Abstract	1
Kurzdarstellung	3
1 Introduction	5
1.1 Specific Capabilities of Humans and Robots	6
1.2 Interactions in Teamwork Among Humans	7
1.3 Contribution	8
2 Background and State of Research	11
2.1 Problem Classes Addressed in this Thesis	11
2.1.1 Concrete Problems	11
2.1.2 Formal Classification of Problem Classes	23
2.1.3 Generalization of Problem Classes and Requirements	30
2.2 State of Research	30
2.2.1 Interaction Modes between Humans and Robots	31
2.2.2 Awareness in Human-Robot Interaction	32
2.2.3 Partially Automated Human-Robot Systems	34
3 The Concept of Situation Overview	39
3.1 Definition of Situation Overview	39
3.2 SO Examples for the Addressed Problem Classes	41
3.2.1 Urban Search and Rescue - RoboCup Rescue	41
3.2.2 Urban Search and Rescue - DARPA Robotics Challenge	42
3.2.3 Robot Soccer	43
3.2.4 Production Hall Logistics	44
3.3 SO in Contrast to other Awareness Concepts	45
4 Communication Concept for Obtaining Situation Overview	49
4.1 Event-based Communication	49
4.1.1 Background: Complex Event Processing	49
4.1.2 Definition of Events	50
4.1.3 Event operators	51
4.1.4 Event Examples for Reference Classes	52
4.2 Event Classification and Tagging	53
4.2.1 Criticality Level	54
4.2.2 Topics	54
4.3 Control of the Event Flow	55
4.3.1 Policies	55
4.3.2 Policy Manager	56
4.4 Discussion Communication Concept	57

5	Interactions between Supervisor and Robot Team	59
5.1	Robot-initiated Queries	59
5.1.1	Queries	59
5.1.2	Query Manager	60
5.1.3	Examples for the Reference Problems	61
5.2	Human-initiated Instructions	63
5.2.1	Application-independent Supervisor Commands	63
5.2.2	Examples for the Reference Problems	64
5.3	Proposed Concept for Realizing Supervisor Commands	68
5.3.1	Background: Task Allocation	68
5.3.2	Modification of Mission Details	72
5.3.3	Influence of Task Allocation – Instantaneous Task Assignment	73
5.3.4	Influence of Task Allocation – Time-extended Task Assignment	75
5.4	Advantages of the Proposed Approach	82
6	User Interface	85
6.1	Background: Interface Design	85
6.2	Interfaces for the Developed Concepts	86
7	Experiments and Results	91
7.1	Simulation of Autonomous Robots	91
7.2	Performance Metrics	91
7.3	Event Detection – Monitoring Trends in a Robot Soccer Match	92
7.4	Human-Robot Interaction	93
7.4.1	Experiments in Urban Search and Rescue - Instantaneous Task Assignment	93
7.4.2	Experiments in Urban Search and Rescue – Time-extended Task Assignment	99
7.4.3	Speed-up Optimization using Queries	105
7.4.4	Application in Robot Soccer - Instantaneous Task Assignment	108
8	Conclusion	111
	Bibliography	113
	Own Publications	123

List of Figures

2.1	Images of a RoboCup rescue arena. From left to right: yellow arena, orange arena, red arena.	13
2.2	A Hector UGV, equipped with several sensors.	15
2.3	Left and middle: The Petman robot, images by Boston Dynamics, Inc. Right: Model of the robot in the DARPA Robotics Challenge Simulator, image by DARPA.	17
2.4	Two scenes from soccer matches in the RoboCup KidSise league.	20
2.5	The Festo logistics competition.	21
2.6	A production hall example: out of 4 types of raw material, 4 different types of chairs can be produced.	22
4.1	Visualization of the interactions among the different components of the event-based communication concept.	57
5.1	The query manager enables the supervisor to dynamically adapt the robots' LOA by selecting different query modes.	60
5.2	Functional parts of a task allocation module.	71
5.3	For instantaneous task assignment, only the tasks and the task costs for each robot are modified by the supervisor commands.	72
5.4	The algorithm to determine the modified task costs based on the original costs according to the supervisor commands.	74
5.5	A simple example showing that cost modification is not sufficient for realizing the supervisor commands when dealing with time-extended task assignment.	75
5.6	For time-extended task assignment, the tasks and the task costs for each robot are modified by the supervisor commands, and additional MILP constraints are added.	76
5.7	Example scenario for joint exploration of a small indoor environment.	82
6.1	An example for a graphical representation of events in a USAR mission.	86
6.2	Functional interface used for the soccer scenario (part 1).	87
6.3	Functional interface used for the soccer scenario (part 2).	87
6.4	Functional interface used for the USAR scenario.	88
7.1	A scene from the semifinal match at RoboCup 2011 between Darmstadt Dribblers and Team Darwin.	92
7.2	Goals scored in the semifinal match at RoboCup 2011 and resulting events.	93
7.3	Arena layout of German Open 2011 and German Open 2012.	95
7.4	Improvements by supervisor support in the USAR mission with instantaneous task assignment.	96
7.5	Small deviations in the robot's localization or perception can cause large errors in the modeled victim location.	96
7.6	Comparison between an autonomous mission and a supervised mission, showing that higher camera coverage leads to more detected victims.	97

7.7	Comparison between an autonomous mission and a supervised mission showing edgy paths due to frequent interactions.	98
7.8	Results for the two-rooms example with calculation times depicted in log scale. . .	99
7.9	Optimal solution for the problem in Figure 5.7 found with planning horizon 9. . .	100
7.10	Real-world experiment with two <i>Ar.Drone 2.0</i> quadcopters.	102
7.11	Routes of the quadcopters in the real experiments. Left: fully autonomous. Right: manual allocation of task number 9.	102
7.12	Computation of a plan and plan costs on a classified elevation map.	104
7.13	Setup for experiment 1 in the large arena.	105
7.14	Example of a query showing the current incumbent.	106
7.15	Incumbents of the optimization for the two rooms example.	107
7.16	A scene from the quarter final at RoboCup 2011, showing the clearing action of the goalkeeper.	109
7.17	Visualization of the role allocation for the simulated clearing action of the goalkeeper.	110

List of Tables

2.1	Classification of the problem classes according to MRTA taxonomy.	23
2.2	Classification of interaction types for the problem classes.	24
2.3	Classification of team coordination for the problem classes.	26
4.1	Resulting event policies after sequential definition of tag policies.	56
5.1	Comparison between the three basic approaches to task allocation.	70
7.1	Number of rows (constraints), columns (variables) and non zeroes (dependency between rows and columns) for the problem in Figure 5.7.	101
7.2	Number of rows, columns and non zeroes for the problem in Figure 5.7 after applying presolve.	101
7.3	Results for supervision experiments with heterogeneous robots in the RoboCup rescue arena.	106
7.4	Different tactics used in soccer matches.	108



List of Abbreviations

CEP	Complex Event Processing
CSCW	Computer-Supported Cooperative Work
DCOP	Distributed Constraint OPTimization
DRC	DARPA Robotics Challenge
HRI	Human Robot Interaction
IMU	Inertial Measurement Unit
LOA	Level Of Autonomy
LRF	Laser Range Finder
MILP	Mixed-integer Linear Program
MRS	Multi Robot System
MRTA	Multi-Robot Task Allocation
OAP	Optimal Assignment Problem
ODE	Open Dynamics Engine
OGRE	Object-Oriented Graphics Rendering Engine
OOI	Object Of Interest
ROS	Robot Operating System
SA	Situation Awareness
SLAM	Simultaneous Localization and Mapping
SO	Situation Overview
TA	Task Allocation
TSP	Traveling Salesman Problem
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
USAR	Urban Search And Rescue
WSN	Wireless Sensor Network



Abstract

For many dangerous, dirty or dull tasks like in search and rescue missions, deployment of autonomous teams of robots can be beneficial due to several reasons. First, robots can replace humans in the workspace. Second, autonomous robots reduce the workload of a human compared to teleoperated robots, and therefore multiple robots can in principle be supervised by a single human. Third, teams of robots allow distributed operation in time and space. This thesis investigates concepts of how to efficiently enable a human to supervise and support an autonomous robot team, as common concepts for teleoperation of robots do not apply because of the high mental workload. The goal is to find a way in between the two extremes of full autonomy and pure teleoperation, by allowing to adapt the robots' level of autonomy to the current situation and the needs of the human supervisor. The methods presented in this thesis make use of the complementary strengths of humans and robots, by letting the robots do what they are good at, while the human should support the robots in situations that correspond to the human strengths.

To enable this type of collaboration between a human and a robot team, the human needs to have an adequate knowledge about the current state of the robots, the environment, and the mission. For this purpose, the concept of situation overview (SO) has been developed in this thesis, which is composed of the two components robot SO and mission SO. Robot SO includes information about the state and activities of each single robot in the team, while mission SO deals with the progress of the mission and the cooperation between the robots. For obtaining SO a new event-based communication concept is presented in this thesis, that allows the robots to aggregate information into discrete events using methods from complex event processing. The quality and quantity of the events that are actually sent to the supervisor can be adapted during runtime by defining positive and negative policies for (not) sending events that fulfill specific criteria. This reduces the required communication bandwidth compared to sending all available data.

Based on SO, the supervisor is enabled to efficiently interact with the robot team. Interactions can be initiated either by the human or by the robots. The developed concept for robot-initiated interactions is based on queries, that allow the robots to transfer decisions to another process or the supervisor. Various modes for answering the queries, ranging from fully autonomous to pure human decisions, allow to adapt the robots' level of autonomy during runtime. Human-initiated interactions are limited to high-level commands, whereas interactions on the action level (e.g., teleoperation) are avoided, to account for the specific strengths of humans and robots. These commands can in principle be applied to quite general classes of task allocation methods for autonomous robot teams, e.g., in terms of specific restrictions, which are introduced into the system as constraints. In that way, the desired allocations emerge implicitly because of the introduced constraints, and the task allocation method does not need to be aware of the human supervisor in the loop. This method is applicable to different task allocation approaches, e.g., instantaneous or time-extended task assignments, and centralized or distributed algorithms.

The presented methods are evaluated by a number of different experiments with physical and simulated scenarios from urban search and rescue as well as robot soccer, and during robot competitions. The results show that with these methods a human supervisor can significantly improve the robot team performance.



Kurzdarstellung

Für viele gefährliche oder belastende Aufgaben wie zum Beispiel Such- und Rettungsmissionen ist der Einsatz von autonomen Roboterteams aus verschiedenen Gründen sinnvoll. Erstens können Roboter Menschen in gefährlichen oder weit abgelegenen Arbeitsräumen ersetzen. Zweitens können autonome Roboter die Arbeitsbelastung eines Menschen verringern im Vergleich zu ferngesteuerten Robotern, dadurch kann ein Mensch prinzipiell mehrere Roboter gleichzeitig überwachen. Drittens können Roboterteams verteilt in Raum und Zeit arbeiten. Aus diesem Grund beschäftigt sich die vorliegende Arbeit mit Konzepten, die einen Supervisor befähigen, ein Team autonomer Roboter zu überwachen und zu unterstützen. Ziel ist es, einen bestmöglichen Weg zwischen den beiden Extremen der vollständigen Fernsteuerung und der reinen Autonomie zu finden, indem der Autonomiegrad der Roboter auf die aktuelle Situation und die Bedürfnisse des menschlichen Supervisors angepasst werden kann. Die in dieser Arbeit vorgestellten Methoden nutzen die komplementären spezifischen Stärken von Menschen und Robotern aus, indem die Roboter die Aufgaben ausführen, die für sie gut geeignet sind, während der Mensch die Roboter in Situationen unterstützt, die er besser lösen kann.

Um diese Art von Zusammenarbeit zwischen einem Menschen und einem Roboterteam zu ermöglichen, benötigt der Supervisor ein ausreichendes Wissen über den aktuellen Zustand der Roboter, der Umwelt, und der aktuellen Mission. Hierfür wurde in der vorliegenden Arbeit das Konzept des Situation Overview (SO) entwickelt, das aus den beiden Komponenten Robot SO und Mission SO besteht. Robot SO umfasst aktuelle Informationen über den Zustand und die Aktivitäten jedes einzelnen Roboters im Team, Mission SO dagegen umfasst den aktuellen Fortschritt der gesamten Mission und die Kooperation zwischen den Robotern. Um SO zu erreichen wird in dieser Arbeit ein neues Event-basiertes Kommunikationskonzept vorgestellt, welches es den Robotern ermöglicht, mit Methoden aus dem Complex Event Processing Informationen zu sammeln und zu diskreten Events zusammenzufassen. Die Qualität und Quantität der Events, die tatsächlich zum Supervisor geschickt werden, können zur Laufzeit angepasst werden, indem für verschiedene Typen von Events Regeln aufgestellt werden, dementsprechend diese gesendet oder nicht gesendet werden. Dies verringert die benötigte Kommunikationsbandbreite, verglichen mit der üblichen Vorgehensweise, weitgehend alle verfügbaren Daten zu schicken.

Der Supervisor kann, basierend auf SO, mit dem Roboterteam interagieren. Die Interaktionen können beidseitig entweder vom Menschen oder von einem der Roboter ausgehen. Die entwickelte Methode für Roboter-initiierte Interaktionen basiert auf Anfragen, die es den Robotern ermöglichen, Entscheidungen an einen anderen Prozess oder den Supervisor abzugeben. Verschiedene Modi zur Beantwortung dieser Anfragen, von voll-autonom bis hin zur reinen menschlichen Entscheidung, ermöglichen es, den Autonomiegrad der Roboter zur Laufzeit geeignet anzupassen. Mensch-initiierte Interaktionen werden auf high-level Kommandos beschränkt, direkte Fernsteuerung wird vermieden, um die spezifischen Fähigkeiten von Menschen und Robotern möglichst gut im Team auszunutzen. In der vorliegenden Arbeit wird gezeigt, wie solche Kommandos in Beschränkungen an die Aufgabenverteilung zwischen den autonomen Robotern übersetzt werden können. Diese Beschränkungen können in sehr allgemeinen Klassen von Aufgabenverteilungsmethoden in das System als Nebenbedingungen eingeführt werden. Dadurch wird die gewünschte Zuweisung von Aufgaben an Roboter implizit erreicht, ohne dass der für die Aufgabenverteilung zuständige Algorithmus tatsächlich Wissen über einen Menschen im System hat. Diese Methode

kann auf verschiedene Algorithmen zur Aufgabenverteilung angewendet werden, z. B. Algorithmen die eine aktuelle Aufgabe für jeden Roboter zuweisen oder mehrere Aufgaben in die Zukunft planen, und zentrale oder verteilte Algorithmen.

Die vorgestellten Methoden wurden in verschiedenen Experimenten sowohl in Simulationen, als auch in Labor-Experimenten mit Such- und Rettungsrobotern sowie Fußballrobotern und auf Roboter-Wettbewerben evaluiert. Die Ergebnisse zeigen, dass die Methoden geeignet sind, um die Leistung eines Roboterteams mit Hilfe eines menschlichen Supervisors erheblich zu verbessern.

1 Introduction

For many dangerous, dirty or dull tasks like in search and rescue missions or factory automation, deployment of autonomous teams of robots is beneficial due to several reasons. First, robots can replace humans in the workspace, which is useful especially in dangerous or narrow areas [12]. Second, *autonomous* robots reduce the workload of a human, because they do not require continuous human attention, in contrast to teleoperated robots, and therefore multiple robots can in principle be supervised by a single human [54]. Third, *teams* of robots allow distributed operation in time and space, and furthermore allow to deploy specialized robots with heterogeneous capabilities, instead of using a single robot that must be able to cope with all different aspects of a mission [97].

Any system design should also account for the existence of errors [88]. Therefore, especially in safety-critical situations, full robot autonomy without any human supervision is not desirable. Pure teleoperation, in contrast, requires at least one human operator per robot (in some cases up to five [80]), and results in a high mental workload, even with extensive operator training [49]. The goal of this thesis is, to develop an approach that combines the benefits of both extremes and avoids their disadvantages. A human in a remote location is enabled to supervise actions of an autonomous robot team, to support the robots in critical situations and to intervene in case of undesired robot behavior, and thus to improve the efficiency and reliability of the mission achievement.

Supervision with a high degree of robot autonomy is chosen for several reasons:

- Simple and repetitive tasks can very well be accomplished autonomously by the robots. When the same tasks are done teleoperated, this would annoy a human operator in the long run, and therefore lead to vigilance decrements and human errors because of inattention, also known as complacency [92].
- Robots and humans have several complementary abilities, and therefore a human can improve the team performance in situations that the robots cannot handle equally well autonomously.
- A human who does not need to continuously interact with a single robot is able to supervise a whole team of robots, and therefore a higher ratio between robots and humans can be achieved than with existing approaches.

For the efficient supervision of autonomous robot teams by a human, four main research problems have to be addressed:

1. *Knowledge base for supervision:* For supervision, a human needs to know other aspects about a robot team than for teleoperation. An appropriate knowledge base provides the supervisor with sufficient information for coordinating a robot team and enables the human to take helpful decisions if the robots need any support, but does not overburden the supervisor with too many details that are not necessary for supervision, and thus prevents information overflow.
2. *Obtaining the knowledge base:* The required knowledge base can be dependent on several factors like the current mission, the current focus, experience and preferences of the supervisor. Furthermore, in real-world applications the available communication bandwidth is

usually limited. Therefore, only required data should be communicated from the robots to the supervisor. Hence, a method is required that on the one hand enables the robots to detect relevant information, but is on the other hand flexible enough to adapt to different setups, user needs, and available communication bandwidth.

3. *Interactions between the supervisor and the robots:* Within the bounds of the interaction role of a supervisor, interactions between the human and the robots shall be enabled. On the one hand, the robots should be able to initiate interactions in situations they cannot resolve autonomously. On the other hand, the supervisor should be able to give high-level commands to the robots to coordinate the team's behavior or to correct undesired behavior of single robots. The robots should be able to integrate these commands into their autonomous behavior, while maintaining coordination with all teammates.
4. *User interface:* The information the robots send to the supervisor needs to be presented in a way that allows the supervisor to quickly assess the situation. Furthermore, the human needs a preferably intuitive way to express the commands to the robots. It needs to be clear to the human that these commands are received and understood by the robots.

1.1 Specific Capabilities of Humans and Robots

Robots can be deployed in applications that can easily be automated, e.g., repetitive tasks, but also in areas where it is not possible for humans to work, either because of too narrow spaces, or because of danger such as collapsing structures or contaminations. However, humans are able to support the robot team from a remote location, to ensure a higher reliability and a better team performance.

To understand why humans can really advance a robot team with few interactions, it is important to be aware of some fundamental differences between humans and robots. In [36], the superiorities of humans among machines and vice versa are discussed. One of the main outcomes is that machines are good in fast routine work, computational power and data storage, while humans' strengths are perception, reasoning, and flexibility. These findings (although over 60 years old!) are in most aspects still valid and can be transferred to a large extent from machines to robots. Also more recent work comes to the conclusion that humans are better in perceptual tasks, solving of unexpected problems, creativity, and imagination, but are inferior to machines regarding precision, accuracy, and remembering details (working memory) [89]. Especially the superiority of humans over robots in problem solving and situation overview is crucial [93], and does not seem to change in the near future.

Although there are several sensors that allow robots to perceive data that humans cannot sense directly (e.g., distance sensors, infrared sensors), humans are much more capable in interpreting data, especially images. A human is even able to recognize almost fully occluded objects and unknown instances of a given object class, which are both very difficult tasks for autonomous image processing [6]. But still, robots are much faster in processing large amounts of data [36]. As an example from the USAR domain, consider the task of generating a 2D map of an unknown environment. Without any autonomy, this requires three humans: one for steering the robot, one for drawing the map, and one for navigation and system overview. With state of the art algorithms for simultaneous localization and mapping (SLAM), robots are able to execute this task faster and much more accurate fully autonomous, without any human intervention, or with only one human responsible for navigation [70].

Repetitive tasks, that have to be executed manually by a human, bring a high risk of operator fatigue and complacency, which means that the human gets bored over time, is less vigilant and can potentially oversee critical situations and make errors [92].

Overall, the many of the capabilities of robots and humans are complementary [36, 89]. Hence, if a human supervisor is aware of the overall situation of the robots, but not necessarily of all details, it makes sense to work with autonomous robots and leave some high-level decisions to the human, who can base decisions on implicit knowledge, situation overview, experience and expertise, which cannot easily be added to the robots' world model. Due to the mainly complementary capabilities of robots and humans, it can be expected that humans can cope well with several of the problems that robots have difficulties in solving autonomously and efficiently. Furthermore, humans often have a better overview about a situation and can therefore intervene by supporting the overall team coordination and high-level task execution.

Also the performance between different humans varies significantly with respect to multitasking and spatial ability [14]. Both are important abilities for supervision of robot teams, because the human has to simultaneously assess the state of the robots and the environment, be aware of the progress of the mission, and detect potential errors and critical situations. Therefore, the interface as well as the duties of the supervisor need to be adaptable to individual human abilities and preferences.

1.2 Interactions in Teamwork Among Humans

When humans work together to achieve a common goal, it is important that every team member is aware about the actions of the teammates. As a basis in social interactions, it is important to acknowledge that a message has been understood, at least by a nodding [89]. A joint activity is defined as a set of actions that are carried out together by a group of people. All team members are required to agree on the joint work, be mutual predictable and directable, and have to maintain a common ground [68]. In human-robot interaction (HRI), the common ground is more difficult to obtain than among humans, on the one hand because of different intelligence levels of the involved agents, and on the other hand because the human needs to understand the robots' functional capabilities [74].

In the area of computer supported cooperative work (CSCW), *workspace awareness* is defined as “the collection of up-to-the-minute knowledge a person holds about the state of another's interaction with the workspace” [53]. It requires knowledge about the identity, location, and actions of the teammates. In particular, it includes information about the objects they are using, the changes they make, but also about what they can see, what they can do, and what they expect others to do. Prinz distinguishes *task awareness*, which contains information about the progress of a specific task, and *social awareness*, which involves knowledge about other people in the workspace [106].

When observing interactions in loosely coupled workgroups [105], similar to the targeted applications, some commonalities can be observed regardless of the scenario, e. g., home care, knowledge work, firemen in a search and rescue scenario, soccer players coordinating with each other and getting instructions from a coach, or people in an office preparing an exhibition at a fair: In all these situations, the overall mission is first subdivided into tasks, that are assigned to the individual team members [80]. Every participant works on some tasks autonomously, and reports the progress to the teammates or the leader, either explicitly by verbal or written communication, or the progress can be directly observed by the others [25]. Whenever someone has problems in

fulfilling a task, a teammate, who is expected to be more capable for this specific problem, is asked for support, or someone who recognizes the problems can offer help or give hints on how to solve the problem [53].

This model assumes a team that is willing to work together and has the achievement of the common mission as highest goal. However, this is a valid assumption in such situations, for humans as well as for robot teams, because the addressed missions are not just collections of arbitrary tasks, but instead provide a common goal, that can only be reached if all tasks are accomplished successfully.

Furthermore, it is important for effective teamwork, that the tasks are shared properly among the team members. The role, the associated expectations and responsibilities of each team member must be clear, otherwise, the performance of the whole team suffers [34].

1.3 Contribution

Inspired on the one hand by the specific capabilities of robots and humans and on the other hand by the way humans interact in groups, concepts are developed to address the research questions described above. The proposed concept considers problem classes, where a team of autonomous robots can be supported by a human supervisor with high-level instructions.

In Chapter 2, the addressed problem classes are described and classified, and the current state of research is summarized.

Regarding the first problem, an adequate knowledge base for a human supervisor, in Chapter 3 the new concept of situation overview (SO) is presented. SO addresses both, aspects related to the robot team and its mission achievement, and aspects related to each individual robot, and thus is more appropriate for interactions between a human supervisor and an autonomous robot team.

The second problem, how this knowledge base can be obtained, is addressed with an event-based communication system, which is described in Chapter 4. Discrete events are detected using methods from complex event processing. Tagging of the events according to semantic topics or criticality levels enables on the one hand different representations and sorting at a user interface, and on the other hand filtering between events, that are not required by the supervisor. The filtering rules can be defined dynamically using policies. This allows to send exactly those events, that are relevant for the supervisor for obtaining SO.

Based on SO, the third problem, how the robots can integrate the supervisor's commands into their autonomous behavior, is addressed in Chapter 5. The interactions between the human and the robots are separated into robot-initiated and human-initiated interactions. For the robot-initiated interactions, queries are used to transfer decisions from a robot to the supervisor. Different query modes, ranging from full autonomy to exclusive supervisor decisions, allow to dynamically adapt the robots' level of autonomy, using tags and policies of the event-based communication system. For the human-initiated interactions, a set of application-independent commands is presented, that allows a supervisor to coordinate a robot team and control the general robot behavior. These commands are translated into modifications to the input data of a task allocation algorithm. The tasks that compose the robots' mission are modified. For instantaneous task assignment, the costs calculated for executing a specific task are adapted. For time-extended task assignment using a mixed-integer linear program (MILP) formulation, the commands are translated into additional MILP constraints.

The fourth problem, the user interface, is discussed in Chapter 6. Common design considerations for designing a user interface are presented, and the application with respect to the developed supervision concepts is discussed. The interfaces used for the experiments in this thesis are however only functional, because the focus of this thesis is on the integration of a human supervisor into an autonomous robot team, and not on the interface.

Some use cases and experiments for the developed concepts are presented in Chapter 7. Most experiments were conducted in simulation because of better availability of heterogeneous autonomous robots and large test environments. Also experiments with real robots are presented.

In Chapter 8 a conclusion is given, and the advantages of the developed concepts compared to available methods are discussed.



2 Background and State of Research

In this chapter, exemplary applications, that are addressed by the developed concepts, are presented. These applications are then classified and abstracted to general problem classes, that can be handled by the developed concepts. Afterwards, the state of research for interaction modes between humans and robots, awareness in human robot interactions, and partial system automation is discussed.

2.1 Problem Classes Addressed in this Thesis

In this section, the problem classes, that are addressed in this thesis, are described. In Section 2.1.1, the concrete scenarios are presented, that are used as exemplary use cases and for evaluation. These scenarios are then classified in Section 2.1.2, using several taxonomies that are focused on different aspects of a human-robot team. Finally, in Section 2.1.3, based on the classification of the concrete scenarios, a general description of the addressed problem classes is derived.

2.1.1 Concrete Problems

The concepts of this thesis can be applied to a variety of fundamentally different problem classes. For demonstration and evaluation, in this thesis the domains of urban search and rescue (on the one hand based on RoboCup rescue and on the other hand based on the DARPA robotics challenge), robot soccer, and production hall logistics are used as reference classes.

Urban Search and Rescue – RoboCup Rescue

Overview

The general environment of an urban search and rescue (USAR) mission is an urban structure, that is partially destroyed, e.g., a collapsed building after an earthquake. The goal is to find trapped humans inside the building as quick as possible, give first aid, and develop plans how the victims can be rescued in a secure, but fast way. To support rescue personnel, human-readable maps of the environment, showing the location of detected victims, have to be provided. As a secondary goal, potential hazards, such as fire, gas leaks or hazardous materials have to be detected, and potentially be included into the evacuation plans.

The application of robots to disaster sites does not aim to replace human rescue personnel or canine units, but rather should provide new possibilities for situations that cannot be accomplished with traditional methods [84]. This includes on the one hand, that robots can operate in areas that are inaccessible to humans (because of narrow spaces, contaminations, and danger of explosions or collapses), and on the other hand that robots can be equipped with sensors that are complementary to the human senses. Examples are distance sensors, that enable the robots to provide accurate maps of the environment for supporting navigation of robots and human rescue personnel, or Geiger-Müller counters for measuring contaminations.

Scenario

So far, the use of robots for real disaster responses is very limited, mainly because of two reasons:

1. *Robot deployments are expensive and require trained specialists.* Most documented robot deployments to real disaster sites were conducted by the Center for Robot-Assisted Search and Rescue (CRASAR). The first deployment was at the World Trade Center (WTC) in 2001. Within 10 days, 8 robot drops could be accomplished, where overall the remains of 10 victims were identified (including findings from the offline data analysis after the actual rescue phase) [12]. Other attempts to deploy robots at disaster sites were less successful. For example, after the collapse of the *Stadtarchiv* in Cologne, there were attempts to use small tracked robots and an active scope camera to search for the two victims [1]. However there were no large enough voids for the robots to enter [2]. The most recent large deployment was in April 2011 in Japan, where the CRASAR team searched for victims after the earthquake and tsunami disaster of March 2011. A list of CRASAR's robot deployments can be found at <http://crasar.org/disasters/>.

CRASAR is not the only group that deploys robots at disaster sites. For example, at the collapse of the *Stadtarchiv* in Cologne, there were attempts to deploy the active scope camera from Tohoku University [56] to support the victim search, but also without success. After the earthquake and tsunami disaster in Japan in 2011, specialists from Tohoku University and Chiba Institute of Technology, Japan, deployed robots to search for victims in the destroyed areas [50]. Unfortunately, the attempts from other groups are not very well documented.

The reports from CRASAR show, that the use of robots at disaster sites is not as easy as many people may think. Although the team does a really good job, there were only 14 robot deployments to disaster sites within 10 years at rather large disasters, were the first response phase and the search for victims lasted for several days. The reasons for that are presumably on the one hand, that the equipment is not available everywhere and has to be shipped to the desired site of operation, and on the other hand, that several trained specialists are required to operate the robots. However, especially in Fukushima, robots were very useful in the phase after the disaster for collecting measurements (e. g., radioactivity or temperature) inside the power plant [76]. Just recently, the first live safe of a robot could be reported after a car wreck in Canada [82]. In summary, the use of robots is difficult and expensive, not only due to hardware cost, but also because very few robots and trained operators are available.

2. *Human rescue personnel is usually skeptical towards new technologies.* An interview with rescue personnel from the German *Technisches Hilfswerk (THW)* revealed, that this skepticism is due to the following reasons: Their highest priority is, to have dependent methods to rescue all victims as fast as possible. New technologies can only be used, if it is clear that they bring a benefit, do not disturb other activities of the rescuers and do not have any other negative effects. This reliability has to be proven in extensive tests, and rescue personnel needs to be trained for the use of the new systems.

Additionally, robots can be deployed to emergency sites when no other approved methods can be used. This can occur either because no resources are available, e. g., because they are needed elsewhere for more important missions, or because they cannot be applied for safety reasons or other restrictions as, for example, no reachable entrances for humans or dogs, that can instead be reached by robotic drones.

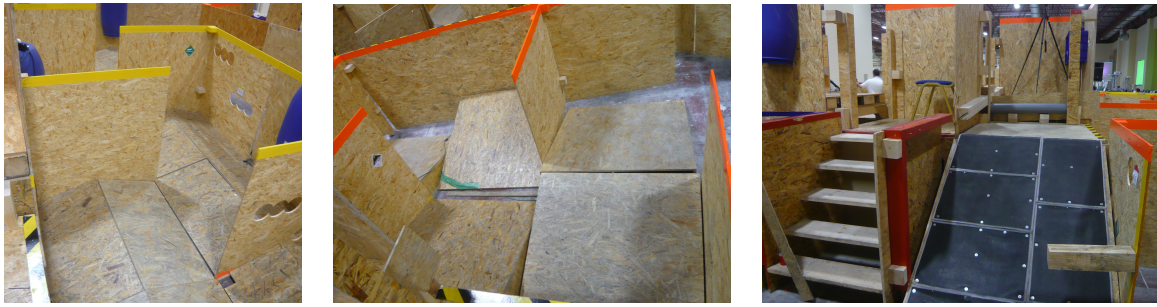


Figure 2.1: Images of a RoboCup rescue arena. From left to right: yellow arena, orange arena, red arena.

A large artificial disaster site, Disaster City, has been built up in College Station, Texas. Disaster City is used as a training environment for canines and human rescue personnel, as well as for rescue robots. It provides training facilities for different types of disasters, as, for example, collapsed buildings, rubble piles or a derailed train. Disaster City is probably the most realistic available training environment for rescue operations.

More repeatable tests are enabled by the reference test arenas for autonomous mobile robots [58], developed by the National Institute of Standards and Technology (NIST). These arenas are used, for example, at RoboCup rescue competitions and are subject to continuous developments. The current arena layout (as of 2013) consist of five different areas, where each part is designed to foster research for specific aspects of the whole problem. The areas are color-coded, to be easily recognizable by humans.

- *Yellow arena:* The yellow arena is designed for fully autonomous robots. The flooring in the random maze is either flat, or consists of moderately inclined continuous pitch and roll ramps (Figure 2.1, left).
- *Orange arena:* In the orange arena, also teleoperation is allowed. The ramps in the maze are steeper than in the yellow arena, and are frequently crossed (Figure 2.1, middle). Also negative obstacles like steps or holes are possible.
- *Red arena:* The red arena features the most difficult terrain (Figure 2.1, right). The maze includes stairs, steep ramps, and full-cubic stepfields. Also here teleoperation is allowed.
- *Blue arena:* The blue arena allows to test manipulation abilities of the robots. At this station the robots can pick up items (as a substitution for health kits or water), and place them inside detected victim boxes.
- *Black/Yellow arena:* The black/yellow arena is also called the *radio dropout zone*. Here, the robots have to navigate autonomously, and the flooring is moderately difficult. It is usually designed as a shortcut-corridor in the orange arena, which should encourage teams with focus on teleoperation to also develop autonomous navigation algorithms.

Although the arenas are not suitable for testing systems for their use at real USAR sites [83], they are very well suited for repeatable tests and research on methods towards more realistic environments. Furthermore, the competitions are a means for developing and evaluating test methods, that frequently become standard test methods for field robots afterwards [59]. Results

obtained in the NIST arenas are comparable, because of the controlled conditions in the standardized environment. It additionally allows researchers to concentrate on specific aspects of a USAR robot, without requiring to also have perfect solutions for all other parts. For example, teams that concentrate on autonomous navigation and mapping do not require a sophisticated mobile robot. Instead, a wheeled robot is sufficient for the yellow arena. In contrast, teams concentrating on developing mobile platforms and teleoperation interfaces do not need autonomous navigation or mapping abilities to score in the orange or red arena. An overview of the RoboCup rescue competition is given in [117].

Several NIST arenas are available all over the world, and they can easily be replicated and simulated. Because of the availability of the arenas, the repeatability and comparability of experiments, the NIST arenas and conditions like at a RoboCup rescue competition are used as reference scenario in this thesis.

Tasks

The overall mission in the RoboCup scenario, finding victims and supporting their rescue, can be subdivided into several tasks.

- *Exploration:* The whole area has to be explored by the robots. This means, that an accurate and complete 2D or 3D map of the area has to be provided. Furthermore, every part of the area has to be searched for potential victims. Whenever a potential sign of life is detected, this results in a new victim hypothesis. Additionally, potential hazards should be located, e.g., by detecting hazmat signs, fire or gas leaks.
- *Victim verification:* Every victim hypothesis, that resulted from the exploration, has to be verified. This requires a robot at first to approach the potential victim, and then verify the hypothesis using arbitrary sensors and algorithms. The artificial victims in the NIST arenas aim for showing the same signs of life as real humans, and hence can be identified by form, heat, motion, sound and CO₂. As soon as the robot has collected sufficient information, it can either present it to a human for final verification, or can autonomously confirm or reject the hypothesis. In the RoboCup competitions confirmation by a human is mandatory.
- *Victim support:* After a victim is confirmed, it has to be rescued as fast as possible. If the victim is able to walk, the robot can guide it to an exit. But usually, human rescue personnel is required to find a way to the victim, which can be supported by the map generated by the robots. Until the rescue personnel arrives, the robots should provide the victim with supplies such as water or first aid kits. In the NIST arenas, this is represented by placing objects from the blue arena inside the victim boxes. Furthermore, if a robot provides a two-way audio connection, humans from outside the hot zone can talk to the victim.

Although the tasks have a fixed order of execution (victim verification always has to precede victim support), they can be modeled as independent tasks, because the subsequent tasks only emerge if the preceding task has a dedicated result. The overall mission is accomplished as soon as the whole search area is explored, all victim hypotheses are verified, and every confirmed victim has received the desired support by the robots. In a competition, the mission is over after a predefined time limit, which is usually 15 or 20 minutes.

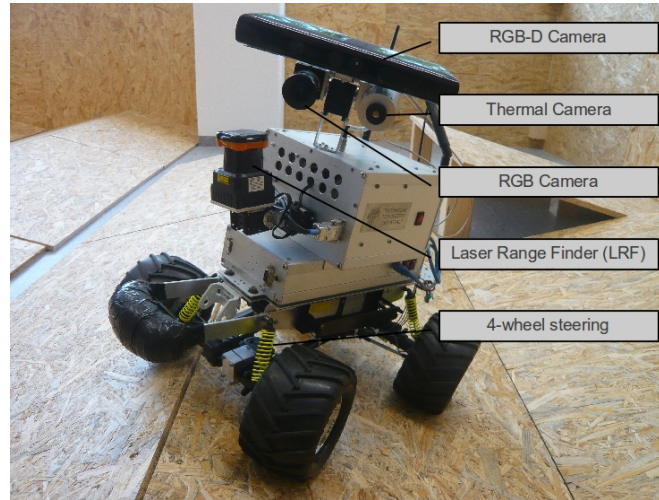


Figure 2.2: A Hector UGV, equipped with several sensors.

Robots

Most common for USAR missions are unmanned ground vehicles (UGVs). In the real-world missions, CRASAR uses different types of tracked robots. The very small ones are used more frequently than the larger ones, because they fit in smaller voids and can therefore be used to search areas inaccessible to humans [12]. From Tohoko University, especially the snake-like active scope camera [56] is used. After the nuclear disaster in Japan 2011, Packbots from iRobot were used to take pictures and measure radiation inside the power plant [52], and T-Hawk drones from Honeywell provided pictures from the bird's eye view from areas that were not reachable by ground vehicles [51].

In the NIST arenas at RoboCup, typically larger robots are used than in the real-world applications. In the red arena, rather large and heavy tracked vehicles are used to negotiate the stepfields, stairs, and ramps. For the orange arena, also all-terrain wheeled vehicles are used frequently. The autonomous robots in the yellow arena are usually wheeled and less mobile than those in the orange and red arena. In the blue arena, the robots need to pick up objects, using a multi-segmented manipulator arm, which is mounted on top of the robot. These larger robots enable to carry more payload, e. g., more different sensors, a versatile manipulator, or more powerful computers. However, also smaller lightweight robots are used, because they are more agile and can navigate through narrower passages. While the robots in real-world missions usually rely only on cameras, most RoboCup robots are equipped with, e. g., cameras, thermal sensors, laser range finders (LRF), RGB-D cameras (D for depth or distance) and inertial measurement units (IMU). Although these robots cannot be deployed directly to real disasters, they support research in many different fields, which can in the future be transferred to fieldable robots. Also unmanned aerial vehicles (UAVs) are used recently, in real-world applications as well as in RoboCup rescue. They range from inexpensive small quadrotors to larger and more expensive helicopters.

In summary it can be seen, that very heterogeneous robots are applied to USAR applications, ranging from tracked or wheeled ground vehicles to different forms of aerial vehicles. Other types, like the snake-like active scope camera or legged robots are not yet very common, but can potentially emerge in the future.

For the experiments in this thesis, robots from the team Hector Darmstadt [4] are used. The wheeled vehicles are equipped with a Core 2 Duo PC, a graphics card for GPU-supported image

processing, and several sensors which are used for mapping, environment modeling, and victim identification (Figure 2.2). The maximum speed on even terrain is approximately 1m/s, but the robot can also negotiate moderate uneven terrain (up to most parts of the orange arena in the NIST arenas). The software is based on the robot operating system ROS (<http://www.ros.org>) [107]. The autonomous capabilities include 2D- and 3D-mapping, exploration, navigation and path planning in unknown environments. Furthermore, a sophisticated world modeling including multi-cue victim detection based on visual, thermal, and range information is provided [77].

Urban Search and Rescue – DARPA Robotics Challenge

Overview

Another different USAR competition has been announced recently as the *DARPA Robotics Challenge (DRC)* under Solicitation Number DARPA-BAA-12-39. This challenge requires a robot to work in a much more unstructured area, which resembles more closely a real disaster site than the NIST arenas.

Scenario

In the DARPA robotics challenge, the robot needs to use structures and tools, that are designed to be used by humans, instead of operating in an environment designed specifically for robots. The challenge will be conducted in two phases. In the first phase, participants will first compete in a simulation environment based on Gazebo (<http://gazebo.org/>), called the Virtual Disaster Response Challenge, which is planned for June 2013, and afterwards using real robots (Disaster Response Challenge #1, December 2013). The second phase will be conducted using real robots, called the Disaster Response Challenge #2, which will be held in December 2014. Because so far the DRC is an unsolved problem, in this thesis it is presented how the developed methods can be applied to this scenario, but no experiments and evaluations are provided.

Tasks

For the DRC scenario, all tasks are known before the start of the mission. In the challenge announcement, the intended tasks are

1. Enter a utility vehicle, drive the vehicle to a target location, exit the vehicle.
2. Travel dismounted across rubble.
3. Remove debris blocking an entryway.
4. Open a door and enter a building.
5. Climb an industrial ladder and traverse an industrial walkway.
6. Use a tool to break through a concrete panel.
7. Locate and close a valve near a leaking pipe.
8. Replace a component such as a cooling pump.

Although these tasks are subject to changes by DARPA, this list already gives an overview about the complexity of the whole mission. Each of these tasks is more difficult to solve and requires a much more advanced and capable robot than the whole RoboCup mission. There will most likely be no fully autonomous solution, instead a closer interaction between the robot and a supervisor or operator (or even a team of humans, acting in different roles) will be required.

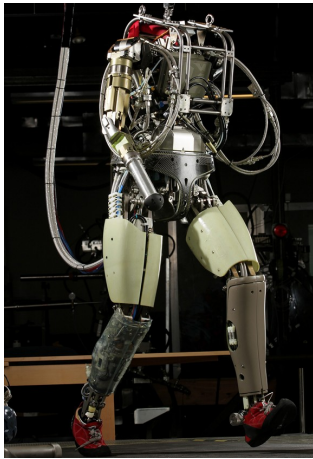


Figure 2.3: Left and middle: The Petman robot, images by Boston Dynamics, Inc. Right: Model of the robot in the DARPA Robotics Challenge Simulator, image by DARPA.

The VRC features only a subset of the above tasks, but are already described in more detail.

1. Walk a short distance to and climb into a utility vehicle, drive along a roadway at no greater than 16 kph (10 mph), climb out of the utility vehicle, and walk to the finish area.
2. Walk across progressively more difficult terrain; for example, progressing from parking lot to short grass to tall grass to tall grass on slope to ditch to rock field.
3. Connect a hose to a spigot and open the spigot by way of turning a valve. This is purely a manipulation task, that is, the robot does not need to travel more than a few steps to the work site.

Robots

In the DRC, the robots are required to use equipment and tools that are originally designed to be used by humans. To solve the challenge, participating teams are free to either develop an own robotic platform, or can use Government Furnished Equipment (GFE), called the GFE platform. The GFE platform will be the humanoid robot Atlas by Boston Dynamics, Inc. (<http://www.bostondynamics.com/>), similar to the Petman robot (Figure 2.3). DARPA makes no restrictions on the design of other developed robotic platforms, in particular, the robots do not need to be anthropomorphic.

In the VRC, a model of the Atlas robot is provided, which can be seen in Figure 2.3, right. It features the same capabilities and interfaces as the real robot will have later on.

Robot Soccer

Overview

In this scenario, two robot teams play soccer against each other. As usual in soccer, the aim is to score as many goals as possible, while hindering the opponent from scoring. The robots can communicate via WLAN. At RoboCup, there are several leagues for different sizes and movement types of the robots (different sizes of wheeled or humanoid robots), for simulated and real robots. In a few leagues, a central planning computer is available, and furthermore an overhead camera delivers the positions of all robots and the ball. In the other leagues, each robot has to localize

itself on the playing field using only on-board sensors, and all planning algorithms have to be executed using the on-board computers of the robots. The number of robots per team and the exact layout of the playing field depends on the concrete league. In the context of this thesis, the humanoid kid-size league is considered.

Scenario

In the RoboCup humanoid league, the playing fields are color-coded: the ground is a green carpet, the ball is orange, the goals are yellow and blue, and yellow/blue landmarks are available. A challenge in humanoid robot soccer is, that the robots can tumble, because of instable motions, contact with other players, or hardware wear. This has to be handled, on the one hand for motion planning to stand up after a fall, and on the other hand for the self localization, because a fall can introduce large errors, especially in the robot's orientation. By the rules, only sensors that correspond to the abilities of a human are allowed. Therefore, the humanoid robots rely on directed cameras for sensing and modeling their environment, and gyroscopes and accelerometers for determining their attitude. In particular, distance sensors like, for example, ultrasound sensors or LRFs, are not permitted to use. In contrast, in the middle size league arbitrary sensors can be used, e. g., omni-directional cameras and distance sensors. In the small size league, an overhead camera and a central planning computer are available.

The robot soccer experiments in this thesis are conducted in the humanoid kid size league. The playing field is 6 m long and 4 m wide. A match lasts two half times of 10 minutes each. Each team consists of 3 players plus two substitutes. During the match, no human is allowed to intervene with the robots, except for taking out robots for service or substitutions.

Tasks

As in a human soccer match, the tasks for each player are described as a role, that describes the behavior of each agent. Unlike a human soccer team, the robots usually change their roles dynamically during the match, depending on the current position of the robots and the ball. The following roles are available in the Darmstadt Dribblers team:

- *Striker*: The striker's main task is to score goals. The behavior can be parametrized by defining when the robot should dribble instead of directly kicking the ball towards the goal. Furthermore, priorities can be defined to prefer passes to teammates or kicks to a free area on the playing field over risky shots on goal if the opponent is blocking most of the goal area.
- *Supporter*: The supporter acts as a backup for the striker. This robot is positioned diagonally behind the striker, because in this position it can on the one hand block goal kicks from the opposing robots, and on the other hand can take over the striker's role, if the striker tumbles or gets dodged by an opponent. The backwards and sideways shift can be adjusted as parameters. This role is derived from human soccer tactics, and aims to increase width and depth in attack and defense [100].
- *Assist Striker*: The assist striker acts as a pass-to power for the striker. This means, that the robot should always be at a position on the field, where it can be reached by a pass from the striker, and from where it can kick towards the goal. If no direct goal kick is possible, the assist striker should at least be in a better position than the striker, i. e., closer to the opponent goal, to bring the team in a more profitable situation.

- *Libero*: The libero is a defender, who is always located close to the own goal. The robot with the libero role positions close to the own goal area, with a sideways shift with respect to the goalkeeper, to cooperatively cover a large part of the goal to prevent or block opponent shots on the goal. The libero's distance to the own goal and sideways shift are parameters of the role description.
- *Goalkeeper*: The goalkeeper is the only robot that is allowed to permanently stay in the penalty area around the goal, and is allowed to touch or pick up the ball using its arms or hands. The main aim of the goalkeeper is to block opponent goal shots by jumping towards the ball. However, if the ball is located in the vicinity of the goal, the robot can also leave the penalty area and kick the ball towards the opponent goal. The size of the clearing area can be defined by role parameters.

The robots are not allowed to tackle each other, and touching the opponent goalkeeper is completely forbidden. Furthermore, it is not allowed that two or more robots stay in the own penalty area for more than 10 seconds (illegal-defender-rule). Likewise, two or more robots are not allowed to stay in the opponent penalty area for more than 10 second (illegal-attacker-rule).

In contrast to the tasks in the USAR scenario, the soccer roles are persistent, i. e., they cannot be completed, instead, they have to be fulfilled as long as they are active. Likewise, new roles can only emerge if the team changes its tactic.

Because only 3 robots are allowed per team at the same time, not all roles can be active simultaneously. Instead, a tactic describes, which roles should be fulfilled, including possible parameterizations for all roles. A very offensive tactic is to use a striker, an assist striker, and a supporter. In contrast, using a striker, a goalkeeper and a libero is a very defensive tactic. Therefore, by varying combinations of the roles, many different tactics can be derived, allowing to address the opponent's strength and tactic.

A tactic can be selected prior to the start and fixed for the whole game, or can be dynamically changed during the game, either autonomously, based on the current game situation, or by human input. Different tactics allow to adapt to specific abilities or tactics of the opponents. For example, while an offensive tactic could work well against a team that cannot kick the ball very far, a defensive tactic could work better against a team that can kick the ball into the goal from any position on the field.

Robots

The robots look very diverse in the different leagues. For example, there are omni-directional wheeled robots in the small size and middle size league, simulated point-models or simulated humanoid robots, and humanoid robots of different sizes in the humanoid league and the standard platform league. However, within one team in a specific league, the robots are usually homogeneous, or only slightly heterogeneous. For example, sometimes the goalkeeper in the humanoid league is equipped with special mechanisms for better blocking abilities or for picking up the ball.

In the humanoid kid size league (the reference scenario in this thesis), the robots are between 30 and 60 cm high, and must have two legs, two arms, and a head. The center of mass must not be too low, the actual minimum height is dependent on the size of the feet. Only sensors that correspond to the senses of a human are allowed. This means, for example, that each robot can have a maximum of two directed cameras in the head, and is additionally allowed to have, e.g., force sensors, microphones, accelerometers and gyroscopes. Therefore, all environmental



Figure 2.4: Two scenes from soccer matches in the RoboCup KidSize league. Robots of the Darmstadt Dribblers playing in cyan.

information, like the robot's own position on the field, the position of other players and the ball, has to be obtained using images.

For the experiments in this thesis, robots from the team Darmstadt Dribblers [3] are used (Figure 2.4). These robots are equipped with one directed camera in the head as only external sensor, gyroscopes, and accelerometers. For motion generation and stability control, a controller-board is used. All high-level modeling, image processing, and behavior control is done on an embedded PC. The high-level software is based on RoboFrame [104].

A team in the humanoid kid size league consists of three players. The robots of each team wear team markers to be easily distinguishable (the equivalent to jerseys). One team wears magenta colored markers, the other team cyan. Colors are changed in the half time break, if the teams do not agree on which team has to wear which color.

In summary, it can be seen that the environment in the soccer scenario is very structured. The field setup is known, including the dimensions of the field, the goals, the artificial landmarks, and the ball. However, the current configuration of the robots and the ball changes very fast, which requires to react quickly and adapt to different situations.

The complete rules of the RoboCup humanoid league can be downloaded at <http://www.tzi.de/humanoid/>.

Production Hall Logistics

Overview

In the production hall logistics scenario, a team of mobile robots has to produce a requested amount of goods out of a given amount of raw material. The overall aim is, that the production process can be fully automated, but without the need to adapt the whole workspace for a specific production process. The environment must be appropriate for robots, i. e., the flooring should be flat, all material and machines have to be reachable for the robots, and the environment should not interfere with the robot's sensors, like, for example, mirrors that can disturb LRFs and complicate image processing. However, the robots should not depend on production lines, rails or other mechanisms designed for fixed work cycles, to enable a more flexible production and allow to reuse the robots and facilities for different products and tasks. Here, only the coordination between robots responsible for transporting the material is considered. The static machines or robots used for, e. g., welding or assembling are disregarded in this context.

This scenario is only an exemplary use case for this thesis. It is used to discuss potential applications of the developed methods in another setup that is fundamentally different to the other scenarios.



Figure 2.5: The Festo logistics competition, image from www.robocup2011.org.

Scenario

The idea of this scenario is similar to the RoboCup Festo Logistics Competition (FLC), see Figure 2.5. In the FLC, only one end product has to be produced. The production process requires in sum 4 operations and consumes 4 pieces of raw material (only one type of raw material is available). The final product has to be delivered to a loading zone, and the processed material needs to be recycled. All material and (intermediate) products are represented by RFID tags mounted on hockey pucks. Therefore, the “production machines” simply change the information stored on the RFID tag. Several redundant machines are available, where each machine is able to execute one specific step in the production cycle. The machines are not perfectly reliable and can be set to “out-of-order” status. A machine’s status is indicated by colored lights on top of the machines.

For the examples in this thesis, this scenario is more generalized. It is assumed that different types of raw material are available, and several different products can be manufactured. Thus, a competition between different products for the raw material can occur if no unlimited resources are available. As an example, consider Figure 2.6. Here, 4 types of raw material are available: wood, cushion, plastic and metal. Depending on how the material is processed, 4 different types of chairs can be assembled: a wooden chair (C1), a cushioned wooden chair (C2), a cushioned plastic chair with metal legs (C3), and a plastic chair with metal legs (C4). While the production of C1 and C4 is completely independent, because they are produced of different material, the production of C2 is in competition with the production of C3, because both require cushion. Similarly, C1 and C2 are in competition for wood, and C3 and C4 are in competition for plastic and metal.

Furthermore, in the FLC, as many products as possible have to be assembled. Here it is assumed, that a specific quantity per product can be requested, associated with priorities or delivery deadlines.

The machines are assumed to be designed in a way that the robots can deploy material and retrieve the processed good. Furthermore, the robots must be able to identify a machine’s current status (ready to deploy, busy, ready to retrieve, out of order). The robots’ actions can fail with a probability $p > 0$. Each machine is able to perform a single production step. For the example in Figure 2.6, one machine can make seats out of wood, but a different machine is required to produce seats out of plastic. For each production step, at least one machine is available. Each machine can fail to work with a certain probability $p > 0$.

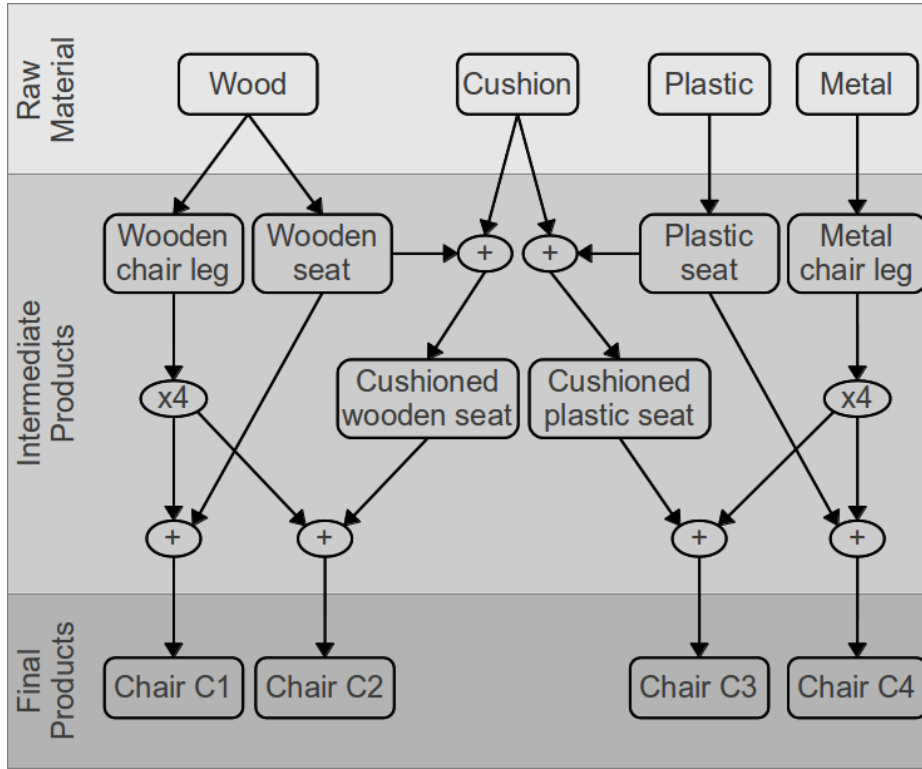


Figure 2.6: A production hall example: out of 4 types of raw material, 4 different types of chairs can be produced.

A storage is available for raw material and intermediate products. The completed products have to be delivered to a designated loading zone.

In addition to the robot team and the static environment, there can be also other moving obstacles, like, for example, humans or other robots. The robots are not required to interact extensively with the humans in the workspace, but must have at least a dependable obstacle avoidance, in order to prevent accidents. Although the collocated humans and robots are not intentionally adversarial, they can block routes or machines, and can therefore disturb the planned operations of the robots.

Tasks

For each product, that has to be assembled, several production steps are required. For the example in Figure 2.6, for a chair of type C3, 4 metal legs and a plastic seat have to be produced. The plastic seat and some cushion are processed further to a cushioned plastic seat. The finished seat and the four chair legs are finally assembled to a chair of type C3. Apparently, some of the productions steps can be achieved simultaneously (e.g., the production of the seat and the chair legs), and some have to be accomplished in a fixed order (e.g., the production of a cushioned seat requires that at first a seat is produced).

In general, the whole mission can be described as a collection of transportation tasks: Raw material or intermediate products have to be brought from the storage to the machines for further processing, intermediate products have to be transported from the machines either to the storage, or directly to the next machine, and final products have to be delivered at the loading zone. Either while modeling the tasks, or when deciding which task to execute next, it has to be considered that enough raw material is available for each product, and that all delivery deadlines are met.

Robots

In the FLC, *Robotino* robots from Festo are used. The robots have omni-directional drives and can be equipped with arbitrary sensors by the participating teams.

For this thesis, no restrictions on the robots are made, except that they need to be adequate for the task. This means, that the robots are mobile in the work space, and have manipulation abilities to pick up and carry material, place it at the machines, and deliver it to the target location. Furthermore, the robots must be able to detect the functionality and the current status of the machines, and recognize the different types of raw material, intermediate products and final products.

2.1.2 Formal Classification of Problem Classes

Taxonomies on Task Allocation

Three taxonomies on task allocation are briefly described, and the problem classes of the previous section are categorized according to these taxonomies. The DRC scenario is not addressed in this section, because task allocation addresses robot teams, while in the DRC only a single robot is considered.

Gerkey and Mataric [44] propose a taxonomy for classifying multi-robot task allocation (MRTA) problems along three axes:

1. *Robots*: Robots are categorized as being able to work on one task at a time (single-task robots, ST) or on several tasks (multi-task robots, MT).
2. *Tasks*: Tasks can either be accomplished by a single robot (single-robot tasks, SR) or require several robots working cooperatively together (multi-robot tasks, MR).
3. *Task assignment*: The assignment of tasks to robots can be planned over time (time-extended assignment, TA) or is done instantaneously (instantaneous assignment, IA)

The problem classes of the previous section are very similar when classified with the MRTA taxonomy. Usually single-task robots (ST) are assumed. Only in the USAR scenario the robots can simultaneously map an area and search for victims. Therefore, if these tasks are modeled separately (which is usually not the case), the robots can be considered as multi-task robots. The tasks for all three scenarios can be fulfilled by single robots and are therefore classified as single-robot tasks (SR). Task assignment can for all three cases be done instantaneously (IA), however, for USAR and production hall logistics, time-extended assignment (TA) is very likely to increase the team performance. For the soccer scenario the fast changes in the environment are usually

Taxon	USAR	Soccer	Production Hall Logistics
Robots	ST	ST	ST
Tasks	SR	SR	SR
Task Assignment	TA/IA	IA	TA/IA

Table 2.1: Classification of the problem classes according to MRTA taxonomy.

not predictable, therefore instantaneous assignment is sufficient. An overview of the classification is given in Table 2.1.

Parker [97] describes the types of interactions between the agents in a team along three axes:

1. *Types of goals: Shared or individual.* The robots can have individual goals, or share the same goal among the team.
2. *Awareness of others: Aware or not aware.* The robots can either be aware of their teammates and reason about their actions, or they are not aware of other agents and treat them as mobile obstacles.
3. *Advance the goals of others: Yes or no.* The actions of each robot can support the other robots' goals or not.

The four most common combinations are categorized into the following classes:

1. *Coordinative:* The robots have individual goals, are aware of others, and their actions do not advance the goals of others. Coordinative behavior is necessary if robots share the same workspace, to avoid conflicting actions of individual agents.
2. *Collaborative:* The robots have individual goals, are aware of others, and their actions advance the goals of others. Collaborative behavior frequently occurs in human working groups, when individuals support others in achieving their goal, without directly working towards an own goal. In robot teams, collaborative behavior can be used to compensate limited capabilities of single entities. In that way, goals can be achieved as a team, that cannot be achieved by the individual robots without teamwork.
3. *Cooperative:* The robots have shared goals, are aware of others, and their actions advance the goals of others. Typical examples for cooperative teams are box pushing and search and rescue tasks.
4. *Collective:* The robots have shared goals, but are not aware of each other, yet their actions advance the goals of others. An example for a collective are (typically biologically inspired) swarms of rather simple robots, which achieve their goals by emergent behaviors.

The classification of the types of interactions for the problem classes of the previous section is summarized in Table 2.2. For all three scenarios, the robots have *shared goals*. In USAR, the common goal is to clear a target area, find victims, locate potential hazards, and provide a map containing all relevant information of the search space. For robot soccer, the common goal is to win the match, by scoring as many goals as possible in a predefined timespan, and hindering the opponents from scoring. In the production hall logistics setup, the robots have the common goal

Taxon	USAR	Soccer	Production Hall Logistics
Types of goals	shared goal	shared goal	shared goal
Awareness of others	aware	aware	aware
Advance goals of others	yes	yes	yes
Classification	cooperative	cooperative	cooperative

Table 2.2: Classification of interaction types for the problem classes.

of producing all requested goods while meeting all delivery deadlines. In all considered problem classes, the robots coordinate their actions and *are aware of their teammates*. In USAR, the robots negotiate the tasks among each other to determine the best applicable robot to clear each search area. Furthermore, the robots need to coordinate their paths to avoid collisions or stuck situations. The soccer robots have to negotiate the role of each player, to ensure, for example, that exactly one robot at a time handles the ball, or exactly one robot acts as goalkeeper. Also here the robots have to coordinate their paths to avoid disturbing each other. The robots in the production hall logistics scenario have to coordinate which robot is responsible for which subtasks (e. g., to transport intermediate products to the right machines, to ensure that all required material is available). Additionally, coordination of machine usage can speed up the production process, because delays caused by waiting can be reduced. Because in all scenarios the robots share the same goals, the actions of an individual robot do *advance the goals of the teammates*. In summary, all considered problem classes are categorized, in terms of interaction types, as *cooperative*.

Farinelli et al. proposed a taxonomy on multi robot systems (MRS) with a focus on the cooperation within the team [32]. The classification of the reference problems is summarized in Table 2.3. The authors classify an MRS along two main dimensions: the coordination dimension and the system dimension.

- *Coordination*: The coordination dimension is subdivided into the dimensions of cooperation, knowledge, coordination, and organization. A *Cooperative* team works together towards a common goal, which is given for all addressed problem classes. *Knowledge* describes whether the robots are aware or unaware of their teammates. In all addressed scenarios it is assumed that the robots are aware of their teammates. Along the *coordination* axis, the authors distinguish between strongly coordinated, weakly coordinated, and non-coordinated teams. Strong coordination means, that the robots explicitly coordinate their actions via a predefined communication protocol, while weak coordination relies only on data exchange without explicit negotiation. The robot teams in the addressed problem classes are assumed to strongly coordinate their actions. Finally, *organization* describes whether the coordination is centralized or distributed. Centralized coordination is subdivided into coordination using a fixed leader (strong centralization) and changing leaders (weak centralization). The soccer scenario is distributed, because no central computer is allowed, and the team composition can change frequently. The controlled environment of the production hall logistics allows to work with a central planning computer, and is hence strongly centralized. In the USAR scenario, distributed organization or weak centralization is beneficial to be robust against communication breakdowns. However, the RoboCup scenario requires a central operator control unit, therefore also here strong centralization is assumed.
- *System*: The second main dimension of the taxonomy, the system dimension, is further subdivided into the dimensions of communication, team composition, system architecture, and team size. *Communication* is categorized into direct and indirect communication. For all addressed problem classes, direct communication is assumed. The *system architecture* is classified into deliberative and reactive systems, and hence describes whether each single agent independently reacts to changes in the environment, or if the whole team reasons (deliberates) upon a reaction to environmental changes. The considered problem classes are not restricted in their system architecture. *Team composition* and *team size* are discussed within the taxonomy on human-robot interaction in the next section.

Taxon		USAR	Soccer	Production Hall Logistics
Coordination	Cooperation	cooperative	cooperative	cooperative
	Knowledge	aware	aware	aware
	Coordination	strong coordination	strong coordination	strong coordination
	Organization	centralized or distributed	distributed	strongly centralized
System	Communication	direct	direct	direct
	Team composition	heterogeneous	homogeneous	homogeneous
	System architecture	deliberative or reactive	deliberative or reactive	deliberative or reactive
	Team size	2 – many	3 – 11	3 – many

Table 2.3: Classification of team coordination for the problem classes.

Taxonomy on Human-Robot Interaction

Yanco and Drury defined a taxonomy on human-robot interaction [128], which spans also several other related taxonomies of different research fields, for example the computer-supported cooperative work (CSCW) time-space taxonomy [27], and the definition of interaction roles [115].

In the following, the different categories of the HRI taxonomy are described, and the classification for the problem classes of the previous section is discussed.

- *Task Type*: This describes the overall objective on a high level, and allows to make implicit assumptions on the setup and the environment. In this thesis, the term *mission* is used instead, because the individual jobs that altogether compose the goal are usually called tasks. For the described problem classes, the task or mission description is urban search and rescue (RoboCup or DRC), robot soccer, and production hall logistics.
- *Task Criticality*: Criticality is a very subjective measure, that describes how important problems or failures in task execution are for the affected people. Task criticality can be rated as high, medium, or low. A task is defined to be highly critical, if a failure can affect the life of a human. Hence, the criticality for a USAR mission (for both RoboCup and DRC) is high, because the environment is dangerous for the rescue personnel and victims, and a robot can even increase this danger if it destroys stable structures. Furthermore, if the robots do not find a trapped victim, the chance to survive decreases drastically. The criticality for robot soccer is low, because a bad system performance only leads to losing a match. For production hall logistics, the criticality is rated as medium, because faulty or missing products can have negative effects for the producing company as well as for the end user, but are usually not life threatening.
- *Robot Morphology*: The robot morphology describes the physical appearance of the robots, categorized into anthropomorphic (human-like), zoomorphic (animal-like), and functional. The appearance of a robot influences how people react to and interact with a robot [37]. The robots in a USAR mission at RoboCup are usually functional, for example wheeled or tracked vehicles. Zoomorphic designs are also subject of current research [108], but are so far only rarely used in real-world disasters, training scenarios or rescue competitions,

therefore the morphology is in general rated as functional. In the DRC is a humanoid robot is used, hence the categorization is anthropomorphic. In robot soccer all three morphology types can be found, e.g., functional (wheeled) robots in the small size league and middle size league, zoomorphic (dog-like) robots in the standard platform league until 2008, and anthropomorphic robots in the standard platform league since 2008 and in the humanoid league. Because the developed methods are applied here to the humanoid kid-size league, the morphology is rated as anthropomorphic. The robots used for production hall logistics are functional.

- *Ratio of People to Robots:* The number of humans and robots is described as a non-reduced ratio, where both numbers can also be described as range, if either the number of humans or the number of robots is variable. In RoboCup rescue competitions, the ratio is usually 1:(1-3). In other experiments the number of robots per human was even higher, up to 1:24 in simulation [114]. Because this work concentrates on supervision of robot teams, the ratio for USAR is given as 1:(2-many). In the DRC, the human robot ratio is 1:1. For robot soccer, a team in the humanoid kid-size league consists of 3 robots. The maximum number of players per team in a human soccer match is 11. This leads to a human-robot ratio for the soccer scenario of 1:(3-11). For the production hall logistics scenario, the human-robot ratio is set to 1:(3-many) The lower bound of 3 is the current number of robots in the Festo logistics competition, and in general in a large factory there is no limit to the overall number of deployed robots.
- *Composition of Robot Teams:* The robots in the team can be homogeneous or heterogeneous, which might have impacts on the applied user interface or communication type. Due to the varying environmental conditions in a disaster scenario, the robots in a USAR team should be heterogeneous. The team can involve wheeled and tracked vehicles, walking or flying robots. This taxon does not apply to the DRC, because there is only one robot. In a robot soccer match, the appearance of the robots is to a large degree specified by the rules, therefore the robots are usually homogeneous, or differ only slightly. For example, a designated goalkeeper robot is often equipped with a mechanism for picking up the ball or for better blocking the goal. It has been proposed to describe heterogeneity on a sliding scale, instead of binary, to be able to distinguish teams with various different types of robots from teams with robots that differ only in few details [7]. Because in general there are only few heterogeneous aspects in a robot soccer team, it is categorized as homogeneous. The robots in the production hall logistics scenario are assumed to be homogeneous, because no tasks arise that require a robot with specialized capabilities.
- *Level of Shared Interaction Among Teams:* This category describes the interactions and coordination among the involved humans (if more than one human is involved) as well as the interactions and coordination among the robots (if more than one robot is involved). In case several humans are involved, they can either coordinate their commands, or send independent commands which the robots have to prioritize. Similarly, several robots in a team can either coordinate their actions, or can all act independently. The value for this taxon is given as a combination of a description for the humans (one human, multiple humans, or human team) and for the robots (one robot, multiple robots, or robot team). The DRC scenario is in the category of one human, one robot. For the other three scenarios, one human is available,

and the robots are coordinating their actions among each other, therefore all are rated as one human, robot team.

- *Interaction Roles:* The categorization of the interaction role the human takes follows the definition of Scholtz [115]. The five different roles (supervisor, operator, mechanic, peer, bystander) are described in more detail in Chapter 2.2.1. This work concentrates on the supervisor role for all scenarios, which is assumed to be the main interaction role. In a USAR mission also an operator can be required to recover a robot from stuck situations. Additionally, in all problem classes situations can occur that require a mechanic to resolve hardware or software failures. However, these situations are assumed to be exceptions, while the focus is here on normal operation mode.
- *Type of Human-Robot Physical Proximity:* The possible values for the physical proximity between humans and robots are avoiding, passing, following, approaching, touching, or none. In the USAR scenario, victims are treated as objects, that have to be detected, but currently not as humans that can interact with the robots. In the other scenarios, no human presence is assumed. Therefore, for all problem classes the physical proximity is classified as none.
- *Decision Support for Operators:* The decision support spans four sub-categories to describe which sensory information is available and how it is preprocessed. One important topic in this thesis is to determine which information should be provided to the human, and how information can be fused to high-level knowledge. Furthermore, in each of the three problem classes different robots can be used, each with different types of sensors. Therefore, only examples for typically deployed sensors and data pre-processing are given, which are not necessarily present in all possible applications.
 - *Available sensor information:* A robot in a USAR mission is usually equipped with a variety of different sensors for navigation (laser range finder, sonar, compass, odometry, GPS, IMU) and victim identification (visual camera, thermal camera, RGB-D camera, CO2-sensor). Lightweight robots, for example unmanned aerial vehicles (UAVs) have a very limited payload and are therefore restricted to a small selection of most important sensors (e. g., a laser range finder and a camera), while bigger and heavier robots often feature as many sensors as possible. The robot in the DRC scenario is equipped with a laser range finder, an RGB-D camera, ground contact force sensors, and sensors for measuring the joint angles and forces. The sensors for the humanoid soccer robots are limited to those that have an equivalent in the human body. Therefore, the robots can have one or two directed cameras, touch sensors, an IMU and other internal sensors to measure, e. g., joint positions, temperature, or forces. The robots in the production hall logistics scenario need sensors for navigation (laser range finder, sonar, odometry, bumper), and for detecting machines and production material (camera, RFID reader).
 - *Sensor information provided to the operator:* In theory, for all problem classes all available sensors can be provided to the user interface. However, this is usually not appropriate, because the human can understand the data much easier if it is preprocessed and visualized in a human-friendly manner. An exception are camera images, which are often easier to interpret for humans than for machines.
 - *Type of sensor fusion:* USAR robots usually fuse several sources of information to learn a map (2D or 3D) of the environment and calculate their own 6D pose within this map.

Furthermore, sensor information to detect signs of life are fused into victim hypotheses. In the DRC, due to the restricted communication bandwidth, as much sensor fusion as possible should be provided. The soccer robots fuse all sensor data to calculate a world model containing the position of all robots (own position, teammates and opponents) and the position and velocity of the ball. The production hall logistics robots fuse information from, e. g., cameras and range sensors, to calculate their position in a given map. Furthermore, camera and RFID information is used to determine the state and location of production material, intermediate and end products in the workspace.

- *Amount of pre-processing of sensors:* In the camera images of the USAR robots, perceptions of potential victims or other hazards can be highlighted. Similarly, in the camera images of the soccer robots perceptions of the ball, landmarks, goals, and other robots can be marked.
- *Time/space taxonomy:* This taxonomy describes whether humans and robot work at the same time and in the same place. It is adopted from [27].
 - *Time:* Interactions can occur either synchronous or asynchronous. For USAR (both, RoboCup and DRC) and soccer, all command from a human should usually be accomplished immediately. For the production hall logistics both scenarios are possible: either a human can on-the-fly change the team’s behavior, or production plans are set up in advance and have to be accomplished later. In this case the synchronous operation is considered, where a human can influence and improve the team behavior during operation.
 - *Space:* Humans and robots can operate collocated or non-collocated. For all problem classes, as already stated for the interaction role and the physical proximity, the interactions are non-collocated.
- *Autonomy Level / Amount of Intervention:* This category describes which percentage of a robot’s task can be accomplished autonomously or requires human intervention. The two sub-categories sum up to 100%. It is not defined, how situations are rated where a human and a robot act simultaneously, e. g., if a robot navigates autonomously while a human monitors the camera images to find victims in a USAR mission. Additionally, it is not clear, how this metric is extended to teams of robots, if for example the values are summed up, or averaged over all robots, and how commands addressed to several robots are included. Furthermore, it is not distinguished between interactions that are necessary for mission achievement on the one hand, and useful, but not required, supporting actions on the other hand. This means that a team can get low autonomy values, although the robots could also achieve the mission more autonomous, but with performance decrements (e. g., slower or less accurate).
 - *Autonomy:* In general, the degree of autonomy is high for all considered problem classes. The robots are able to solve all tasks autonomously, but probably less reliable then with human support. Due to the criticality, in USAR missions a closer integration of the human is desirable to safeguard important decisions or critical tasks, whereas in robot soccer and production hall logistics the human intervention is mainly to coordinate the team and specify strategies to reach a goal. An exact value for the percentage of autonomy is not given here for the above mentioned open problems for evaluating the correct value.

-
- *Interaction*: As the robots have a high degree of autonomy, the interaction time between the human and each single robot is very low. Additionally, there are on the one hand necessary human interaction, where the robots need support to fulfill a task, and on the other hand situations where a human can improve the team performance, but is not required to interact. Hence, the interaction value is (like autonomy) a range, but an exact value cannot be given here for the reasons described above.

2.1.3 Generalization of Problem Classes and Requirements

According to the taxonomies on task allocation, all addressed problem fall into the classes of ST-SR-TA or ST-SR-IA: Robots are considered that work on one task at once, and the tasks are considered to be modeled in a way that they can be achieved by a single robot. The task assignment can be either time-extended or instantaneous, depending on the current application, and available infrastructure for computation and communication. All considered applications assume a *cooperative* robot team, which means that all robots share the same goal, are aware of each others, and advance the goals of their teammates with their own actions. The robots are assumed to communicate directly, to achieve a strong coordination. This coordination can be achieved using centralized or distributed algorithms.

The general team composition consists of a human, acting in the role of a supervisor, and a team of homogeneous or heterogeneous robots (only one robot in the DRC scenario). The interactions between the human and the robots are all remote, the existence of collocated humans is not assumed, and hence no physical human-robot interaction is considered. All interactions are on a high level and affect the robots' plans and goals, but not the lowest action level corresponding to teleoperation. The robots have a high degree of autonomy, and are even able to solve their tasks without any human support.

The criticality of the targeted applications varies from low to high. Also, the structuring of the environment is different for all scenarios. The applications of soccer and production hall logistics assume a very well-defined and structured environment, while the environment in a USAR scenario (both, the RoboCup application and the DRC) is often very unstructured and cluttered, and the environment is not necessarily known in advance. In the soccer scenario, mobile elements apart from the own robot team are present, that can move very fast and can not in all aspects be controlled by the robots, for example, the ball and opposing robots. The other scenarios are assumed to be static except for changes made by the robots themselves. The available communication bandwidth, for interactions between the supervisor and the robots as well as for the communication between the robots, is usually very limited.

Other applications, that fit within the same classes as the example scenarios, and are therefore also potential application for the developed concepts, are, e. g., a fleet of service robots, that have to clean an environment, hazardous waste cleanup [95], cooperative multi-robot observation of multiple moving targets (CMOMMT) [96], or distributed sensing [22, 124].

2.2 State of Research

In this section, an overview about the state of research is given for interaction modes between humans and robots, and for different awareness models used in human-robot interaction. Afterwards, an overview is given about design considerations for partially automated systems, and some examples are presented where partially autonomous robots interact with a remote human.

2.2.1 Interaction Modes between Humans and Robots

Frequently, the term *Human Robot Interaction (HRI)* is associated with the interactions between one human and one robot, either with a remote interface, or by physical interactions, speech or gesture recognition. However, interactions can also occur between several humans and one robot, one human and several robots, or several humans and several robots. If more than one human or more than one robot is involved, coordination is mandatory to prevent on the one hand contradictory commands from two humans to a single robot and on the other hand two or more robots disturbing each other due to a lack of agreement on available resources or space.

Interactions between robots and humans can occur on various different levels. Scholtz [115] identified five different roles a human can perform: supervisor, operator, mechanic, peer, and bystander. The modes of interaction are entirely different for all five roles, and hence the human needs to have a specific understanding of the current state of the robots and the world, depending on the interaction role.

- *Supervisor*: The supervisor monitors and controls the overall situation, the commands are given on a high level, e. g., by defining goals and modifying plans. A supervisor needs to have an overview of the situation, which includes knowledge about the current goal and status of each robot, but does not need to be aware of every detail in the robots' environment. A single supervisor is usually able to interact with several robots simultaneously.
- *Operator*: The operator intervenes at the robots' action level, which is also known as teleoperation. This means, that the operator can either specify the actions the robots should carry out autonomously, or the operator can directly control a robot's motors, or any level of autonomy in between these two extremes. These kinds of interactions require the operator to have a very detailed idea of the robot's current status and its environment, to be able to predict the consequences of an action. Usually, an operator cannot interact with more than one robot at a time.
- *Mechanic*: The mechanic can physically change a robot's hardware or software. He needs to know which parts of the robot's behavior failed and how, to be able to adjust hardware or software. The mechanic can only interact with one robot at a time.
- *Peer*: The peer interacts with the robots on a higher level than the operator, but stays within the bounds defined by the supervisor. Interactions are usually collocated and have the goal to accomplish a task together with a robot. The peer needs to know about the robot's current world model, and what actions the robot can carry out. A peer can interact with one or more robots simultaneously.
- *Bystander*: The bystander is collocated with the robots and can only cause the execution of a subset of the robot's possible actions, e. g., obstacle avoidance. The bystander needs to be aware of the robot's capabilities and current plans, to be able to understand the intention of the robot's actions. A bystander cannot really control a robot, but only shares the same workspace. Most important is that robots and bystander do not disturb each other. Also the bystander can interact with several robots simultaneously.

For the human supervisor role, Sheridan identified five main tasks, that are usually performed subsequently [118]: *planning* what the agents are supposed to do, *teaching* the plan to the agents,

monitoring the autonomous behavior, *intervening* if the actions are not carried out as desired, and *learning* from the interaction cycle, to be able to improve the performance in the future.

The addressed scenarios (c.f. Chapter 2.1) all assume to involve a robot team and a remote (not collocated) human. Therefore, the roles of mechanic, peer, and bystander are disregarded. Wang and Lewis showed for an example of a team of three robots in a USAR environment, that a mode between pure teleoperation and full autonomy promises to end up with a better performance than both extremes [126]. By means of the different capabilities of humans and robots (Chapter 1.1), it gets clear that the role of a supervisor is best suited for interactions that make use of the strengths of both humans and robots. Operator interactions can be used as backup, if the autonomous mode with supervisor support fails. However, the optimal level of autonomy still needs to be worked out, i.e., which parts of a task should be carried out with human support, and which tasks should be accomplished autonomously by the robots.

2.2.2 Awareness in Human-Robot Interaction

For human-robot teams as well as for humans working with partially autonomous software agents, the information the human has about the agents' state and actions is an important factor to enable good team performance [110]. This awareness is crucial for agents of all autonomy levels, ranging from fully teleoperated to fully autonomous agents. However, the required information is dependent on the mission and the interaction role the human performs.

For teleoperated robots, in real world missions [84], realistic training environments [110], as well as in artificial competition arenas [26], most navigational errors and collisions with obstacles are due to an insufficient situation awareness. For automated systems, Norman states in [88], that the system should continuously provide feedback about its activities, regular and unanticipated events, so that the supervisor knows the system is working, and can detect potential problems early, instead of suddenly being confronted with a hazardous situation, that the system cannot handle autonomously.

These examples highlight the importance of an adequate awareness of the human about the supervised systems, which includes not only information about critical incidents, but also a system's regular status. The exact composition of information, i.e., what is relevant and what is not, is to a large extent dependent on the current mission of the robots, and dependent on the human's interaction role and the current tasks within the bounds of this role.

Most commonly used is the term *situation awareness (SA)*, which was defined by Endsley as “*the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future*” [30]. To achieve SA, a human has to pass three levels:

- *Level 1 SA* is the perception of status, attributes, and dynamics of relevant elements in the environment,
- *Level 2 SA* is the understanding of the meaning and the correlation of the perceived level 1 elements,
- *Level 3 SA* is the projection of the current state into the future.

Level 2 and 3 can usually be achieved easier by experienced users than by novices, because they can base their knowledge on previous experiences and already have developed detailed mental models of the system [30].

Especially in the USAR domain, good SA is usually associated with a complete knowledge about the status of a robot and its surrounding. This includes the robot's health (e.g., the functionality of all sensors, the battery status, software failures, mechanical breakdowns), the robot's direct surrounding (e.g., the structure of the terrain, nearby obstacles, objects of interest such as victims or hazards, other robots or humans in the vicinity), and the relation between the robot and the environment (e.g., the robot's position with respect to a fixed coordinate system, the robot's tilt, pan, and yaw angle with respect to a fixed coordinate system, the distance between the robot and the obstacles, maneuverability of the robot on the given terrain). This information is commonly gathered by full video streams of the cameras or live map-data, which produce a large communication overhead.

In real-world USAR missions, the operators gain situation awareness by interpreting raw sensor data. Here, situation awareness "*is primarily about spatial relationships between objects and how that impacts robot navigation and coverage of the void*" [84]. The perception of the sensor data corresponds to level 1 SA, the derivation of spatial relationships is level 2 SA, and the impacts on robot navigation and coverage of the void correspond to level 3 SA.

However, while this type of SA is well suited for direct teleoperation, it is not applicable for supervisor interactions, that occur on a higher level, and hence require a different type of knowledge of the situation. For example, an operator may have to control the robot's motors for maneuvering the robot to a specific position, whereas a supervisor can simply give the command "go to position xyz". In this situation, the operator needs to know the robot's distance to the surrounding obstacles, the robot's exact position, and the possibilities to move the robot. For the supervisor, it is enough to roughly know the robot's position, and if the robot is able to reach the desired position. Everything else has to be handled by the robot autonomously.

Because the definition of SA is derived from the egocentric SA of a pilot in an aircraft, also SA for a robot operator is egocentric from the point of view of a single robot. A generalization to a whole team of robots is difficult, because usually a human cannot track such detailed information for many robots simultaneously. Instead, the presentation of all SA elements of many different robots to a single operator can quickly lead to information overflow [84]. Riley and Endsley propose to have operators and payload specialists (responsible for interpreting sensor data), either one for each robot in the team or for a small group of robots [111]. The (single) human mission controller communicates only with the payload specialists, and therefore receives only SA elements that are filtered by humans, which makes it much easier to maintain team SA.

As another model of awareness within human-robot teams, Drury et al. set up an HRI awareness framework, with a focus on the role of the operator [26]. They identify five components:

1. *Human-robot awareness* describes the knowledge of the humans about the robots' locations, identities, activities, status and surroundings, and the certainty of this knowledge.
2. *Human-human awareness* describes the understanding a human has about the other humans' locations, identities and activities.
3. *Robot-human awareness* is the knowledge a robot has about the commands and any authority constraints imposed by the humans.
4. *Robot-robot awareness* is a robot's knowledge about commands from other robots, and the other robots' current plans and tasks.
5. *Humans' overall mission awareness* describes the humans' overview of the activities in the team and the teams' progress towards the mission goal.

They define the term *HRI awareness violation* as “*HRI awareness information that should be provided [but] is not provided.*” The quality of HRI awareness is rated based on the number of HRI awareness violations, that cause faulty operator behavior. Every violation is traced back to one of the five awareness types.

A specific type of awareness violation is disorientation, i. e. the human does not know the robot’s position and orientation. Disorientation occurs frequently in realistic environments [110, 84], but only rarely in structured artificial environments [26]. A reason for this discrepancy may be on the one hand the more difficult environments, and on the other hand the available sensor data: Most robots, that are used in the NIST arenas, provide distance sensors (sonars, laser range finders) and mapping abilities, which can help to orient in an unknown environment, whereas the interfaces in [110, 84] rely only on camera data. The camera images are much harder to interpret for humans than range data or maps, because they are usually mounted at low height and therefore show perspectives that are unusual for humans, especially if the environment looks very uniform, as in USAR environments [111].

For the scope of this work, human-human awareness and robot-robot awareness are of secondary importance, because they are not directly affected by supervisory control. Robot-human awareness is important to enable the robots to transfer critical decisions to the human, which will be covered in Chapter 5.1. Furthermore, in Chapter 5.3 the robot’s awareness of the supervisor commands is addressed. The humans’ overall mission awareness is crucial, especially for a human supervisor. However, the definition is rather loose, examples in [26] are given either for single robots, or a small number of successively teleoperated robots, and hence it is not clear, if this type of awareness is adequate for supervising a robot team. Presumably, it does not cover all required information, but is rather appropriate for monitoring the general mission progress, independent of the team members, while disregarding the health and state of individual robots. Human-robot awareness is very similar to Endsley’s SA. It requires the human to have a very detailed model of each robot, its position and orientation, the status of all sensors and actuators, the robot’s surrounding, including the distance to nearby obstacles in all directions, soil properties, and the robot’s possibilities to move in this specific situation. This type of awareness is adequate for an operator, but is much too detailed for a supervisor, because on the one hand, the supervisor does not need this large amount of information, and on the other hand also cannot maintain such a detailed model for many robots in a team.

For supervisory control of robot teams, the detailed SA or human-robot awareness contain far too much detailed information, which can lead to information overflow [84]. Drury’s *Humans’ overall mission awareness*, in turn, only contains the overall mission progress, without taking into account the activities and status of each team member. Hence, these definitions cannot provide an adequate basis for high-level interactions between a human supervisor and a robot team.

2.2.3 Partially Automated Human-Robot Systems

In general, the research area of human-robot interaction (HRI) covers all interaction roles of Chapter 2.2.1. Here, only the cases relevant for this thesis are considered, especially the large area of physical HRI is disregarded here. The focus is on robot teams, that are to a large extend autonomous, and a human supervisor can interact with this system.

Partially Automated Human-Robot Systems – General Design Considerations and Challenges

In the general case, when some parts of a system (as, for example, a robot team) are automated, while other parts are under manual control, several issues have to be considered, that do not occur in solely manual controlled or fully automated systems. These issues are on the one hand related to the way humans use automation, and on the other hand arise because during system design side effects related to *human factors* are not considered [16].

Parasuraman and Riley state, that *if* and *how* a human uses automation is influenced by several factors, including trust in the automation, confidence in the own capabilities, mental workload, and individual differences [93]. These factors are also dependent on the actual performance of the automation.

However, automation can also be used wrong, leading to performance decrements, instead of the desired improvements. In [93], three types of wrong use of automation are distinguished: misuse, disuse, and abuse:

- *Automation misuse* is defined as “*overreliance on automation*”, i. e., the automation is used more than it should be used, leading to decreased performance or even accidents due to a loss of situation awareness. System designers have to be aware of potential automation misuse, to be able to anticipate these situations and think about solutions to prevent misuse, e. g., by providing more salient feedback to the human about the internal system state. This effect is also known as over-trust or complacency [94].
- *Automation disuse* is “*underutilization of automation*”, e. g., alarms, safety systems or other automated parts are switched off (often because of too many annoying false alarms), although they could increase the performance or enhance system safety. System designers must consider the decision threshold of an alarm, as well as the a priori probability of a hazardous event, to be able to set up a system that does not result in human mistrust, and hence disuse.
- *Automation abuse* is defined as “*inappropriate application of automation*”, which means that system parts are automated without taking into account, if this actually improves the whole system (including human) performance. In the worst case, accidents occur due to automation, that could have been prevented by a human. These errors can in turn cause automation misuse or disuse.

The authors conclude, that “*many of the problems of automation misuse, disuse, and abuse arise from differing expectations among the designers, managers, and operators of automated systems*” [93, p. 249].

Especially appropriate feedback is a crucial part to prevent automation misuse. Automated systems, that do not direct the attention of the human properly, or do not report all relevant information, lead to a loss of situation awareness, which can in turn cause wrong decisions of the human [68]. This is consistent with Norman’s statement, that feedback from the automated system is essential to avoid that the human is suddenly confronted with errors, that could have been anticipated, if the human had been kept in the loop [88].

A concise overview of problems that can arise with supervisory control is given by Chen et al. [16]. They identify the following critical areas for supervisory control:

- *Operator multitasking performance:* A human’s monitoring performance decreases, if in parallel other tasks have to be executed. Also the reliability of the automation and the task complexity influence this performance decrement.
- *Trust in automation:* Trust is an important factor, because humans usually interrupt the automation, if they believe to perform better manually. However, humans tend to either over- or under-estimate their own capabilities, and also the reliability of the automation is often perceived wrongly. Together, this leads to a wrong assumption about the relation between automation performance and operator performance, and hence causes automation disuse or misuse.
- *Situation awareness:* If the human is not actively engaged in the execution of a task, a lack of system feedback and task switching can have negative effects on the operator’s situation awareness. A low situation awareness can lead to wrong decisions of the operator.
- *Operator workload:* The perceived workload influences how automation is used, i. e., other levels of automation are used in high workload situations compared to low workload situations. The operator’s workload increases, if trust decreases, because the system has to be monitored more carefully.

In addition to the above effects of automation, also skill degradation can occur [94], if an operator does not have to fulfill specific tasks for a long time. This implies that, in case the automation fails, the human is potentially not able to solve the task manually. This problem can be overcome, for example, by regular operator trainings or by occasionally letting the operator perform tasks manually, instead of using the automation.

For deciding which parts of a system to automate, and which parts should be accomplished teleoperated, Wampler suggested to apply teleoperation if tasks are not repetitive, if the workspace is not safe for using industrial manipulators, if the tasks require manipulation abilities, including hand-eye coordination, if the tasks require scene understanding, object detection, or other sophisticated perceptual abilities, if teleoperation is possible with the available communication channels, and if a trained operator is available [125]. However, these suggestions are from 1990, when autonomy was not yet well explored.

More recent work suggest to choose an automation level for different aspects separately [94, 111]. The automation level can range from full teleoperation, over assisted teleoperation, autonomy with human assistance, to full autonomy. For selecting an autonomy level, not only the human performance should be taken into account, but also the automation reliability and the costs of faulty behavior [94]. In general, the capabilities of humans and robots should be taken into account when determining the autonomy level, instead of simply automating everything that is possible [93, 87]. Experiences from real-world deployments showed, that especially the cognition should be supported by intelligent algorithms, because the human operator suffers from fatigue due to a lack of sleep, which leads to errors in operator perception and control [12]. This shows that the requirements on a system can vary with different setups, in this case, between laboratory conditions and deployments in real applications.

Based on joint activities among people, Klein et al. discuss challenges to be addressed by joint human-agent activities [68]. They state that agents, to be efficient team players, must follow the same goals, be aware of the actions of their teammates, make their actions understandable to their teammates (including humans), be directable, negotiate goals, and support the human in managing attention. Johnson et al. define the principle of coactive design, where it is assumed

that humans and agents need to be aware of interdependent joint activities, instead of making agents completely independent by increasing their level of autonomy [60]. This corresponds to the goals of this thesis, that a supervisor needs to be aware of the actions within the team, and that interactions can be initiated by both, supervisor and robots.

Realizations Examples for Human-Robot Teams

Louizou and Kumar extend a controller based on almost asymptotically input-to-state stability navigation functions to accept human inputs, while guaranteeing properties of the original controller like convergence and obstacle avoidance [75]. This concept is useful for overwriting a robot's autonomy for a short time period, but is not appropriate for controlling several robots simultaneously.

Approaches that allow a single supervisor to deal with robot teams and do not require continuous high bandwidth communication can be found in the area of sliding autonomy or mixed initiative. The easiest way to allow a single human to control a team of robots is to sequentially operate each single robot, while neglecting the other robots, e.g. [63, 122, 45]. With this concept, the number of robots that can be controlled by a single human is restricted by the performance of the robots' autonomous behavior. The interface presented in [39] allows to control either a single UAV, while the rest of the team maintains the formation requirements, or to give directions to a whole formation. Here, the human cannot give instructions that violate the formation constraints of the team. This implies that the approach does not cover teams where the robots work on independent subtasks.

In [116], Markov models are used to decide whether a robot works on a task autonomously or is being teleoperated by an operator. This requires continuous communication connection only during the teleoperation phases. During these periods the supervisor cannot attend to the actions of the other robots in the team. The system proposed in [123] allocates the control over a robot to either an autonomous agent or a human operator based on the expected performance. Critical situations can be detected, if the autonomous performance drops below the expected performance range. In [11], different autonomy modes for a tracked vehicle are presented, that allow to chose a mode that is appropriate for a given situation. Also the mixed initiative system presented in [48] allows the operator to manually switch between autonomy modes, where the operator input varies from goal input to full teleoperation. The agents may act autonomously within the authority bounds of the current autonomy level. Similarly, in [126] the operator can assign waypoints, move the camera, or fully teleoperate a robot, while the robots perform the remaining tasks autonomously. In the experiments, the mixed initiative approach outperformed both, the teleoperated as well as the fully autonomous robots. Also the experiments of Parasuraman et al. lead to the conclusion that adaptive automation outperforms full teleoperation or fixed autonomy levels [91]. Also in [85], augmented autonomy allows the operator to assign waypoints to the robots, otherwise, the robots select their next goal autonomously. Results show, that these methods are appropriate to deal with a larger number of robots and can produce much better results than purely autonomous or purely teleoperated systems. However, they still require periods of continuous communication connection between the operator and the robots, and most of them can hardly be extended to fundamentally different scenarios, where the main focus is not on search or exploration.

A completely different approach is described in [38], where the robots can ask questions to the human supervisor. Similarly, in [67], the human is treated as a source of information for the robots. The level of autonomy is controlled by adjusting the cost to contact the supervisor. The

blended teleautonomy system presented in [127] enables the robots to detect situations where human intervention is helpful, which are in this context the states of robot stuck, robot lost or victim found. A mediator evaluates the safety of the inputs from the human, and can also refuse the execution of risky commands. Human supported decision taking is presented in [18], here two variants are proposed: management-by-exception, where the operator can veto against an autonomous decision, and management-by-consent, where the operator needs to confirm an autonomous decision before execution. In [8], policies are used to restrict the autonomy bounds of the robots, in this context also rules are defined about which messages the robots are required to send to the human. These approaches are promising to be applicable to larger robot teams in real-world environments, because they do not require continuous human attention to a single robot and require less bandwidth as they do not rely on video streams. However, they are still not very flexible to be adapted to fundamentally different scenarios or for on-line adaption to different operator preferences. Furthermore, the events that require operator intervention are detected manually, and yet no method has been provided to flexibly detect complex events in arbitrary complex situations.

In the Asymmetric Broadcast Control (ABC) architecture, the state space of a robot swarm is reduced to generalized state space which is independent of the number of robots in the team. This allows a supervisor (human or agent) to control abstract attributes of the whole team, like shape, position and orientation of the formation, instead of controlling each robot individually [78]. However, this method cannot be transferred to robot teams performing independent sub-tasks. Chen et al. present an intermediate layer between a human operator and the robot team called RoboLeader, which interprets commands from the human and translates them into control commands for the robots with less capabilities [15]. Results show, that this system improves the performance for human operators with high spatial ability, but has no effect on the performance of operators with low spatial ability. Hardin et al. present a mixed initiative system, that assumes large teams under human control, sharing autonomy dynamically [54]. Essential to this approach are the complementary abilities of humans and robots, the capability of the robots to advance to the mission goal even without human intervention, and the possibility for the operator to interact efficiently with large numbers of agents. Few et al. describe a collaborative tasking mode, that takes away some authority from the human, and lets the robots execute parts of the tasks autonomously, which reduced the human workload and operator confusion in the experiments [35].

The discussed approaches are specifically designed for a given scenario, and do not support arbitrary supervisor commands, which are independent of the concrete application scenario.

3 The Concept of Situation Overview

For any supervisor or team leader it is crucial to obtain an appropriate model about the current state and activities of the team members. This holds for human-robot teams (c.f. Chapter 2.2.2) as well as for human teamwork and human leaders of teams of humans (c.f. Chapter 1.2). However, commonly used awareness concepts as presented in Chapter 2.2.2 are not in all aspects appropriate to be applied to supervisory control of robot teams: while SA demands for a very detailed knowledge about a robot and its surroundings, which cannot be obtained for a whole team of robots because of too many information to keep track of, the notion of mission overview only contains information about the mission progress and completely disregards the status of the individual team members. Awareness concepts that are applied to human teamwork don't have to deal with the inherently different capabilities of humans and robots, and are therefore also not appropriate for human supervision of robot teams. To overcome this gap, the new awareness-concept of *situation overview* (SO) is introduced in this chapter. The idea of SO has first been published by the author in [102], and has been presented in further detail in [101].

SO is an important prerequisite for the interaction role of a supervisor. It can be applied to fundamentally different problem classes (like the ones described in Chapter 2.1.3) and is therefore not bound to any specific scenario. In particular, in this chapter examples are given for the reference problem classes described in Chapter 2.1.1. Furthermore, commonalities and differences between SO and other awareness concepts are discussed.

3.1 Definition of Situation Overview

SO is defined as the combination of *mission SO* for the whole team and *robot SO* for each robot in the team.

- *Mission SO* is the supervisor's knowledge about the mission progress in general, and the knowledge about the current and planned actions of the robots to contribute to the mission success. It furthermore involves knowledge about the coordination among the team members, i. e., if the robots take into account the plans of their teammates for their own behavior.
- *Robot SO* describes the supervisor's understanding about a robot's status and environment, its current and planned actions, including the robot's ability to accomplish the current task. This includes the knowledge about a robot's health, i. e., if the robot is working correctly or if it needs support from either the supervisor or from an operator.

Robot SO is related to each robot's individual performance, whereas mission SO is related to the team performance. This distinction is important because the team performance is *not* the sum of the individual performances. As an example, consider a team of robots exploring an office building. On the one hand, if every single robot would follow the same deterministic exploration strategy, all robots would cluster in the same place without coordinating their activities. In this case, even though each team member may show a good performance, the team performance is poor, potentially even worse than the performance of a single agent, because the team members disturb each other. On the other hand, in a well-performing team that coordinates properly, a single malfunctioning robot may not be recognized if the individual robot performance is disregarded.

and only the team performance is considered. This shows that mission SO and robot SO are complementary and together compose a complete SO.

Mission SO is in particular defined as the combination of the following topics:

- *Mission achievement* includes knowledge about which parts of the mission are already completed, and what still remains to be done.
- *Mission progress* refers to the expected trend of the remaining tasks of the mission. The supervisor needs to know if all open issues can be handled by the robots, and in case there are any time constraints, if these tasks can be achieved in time.
- *Team coordination* addresses the teamwork among the robots, i. e., if the robots coordinate their actions to avoid interferences and to achieve the best possible team performance. Especially, the assignment of tasks to robots should respect each robot's capabilities and current status.

The individual topics of robot SO are defined as follows:

- *Robot health* describes if a robot is functioning correctly or if there are any defects (e. g., concerning sensors or actuators). If a defect occurs, the supervisor needs to be aware of the consequences for the robot's capabilities, and should know if the defect can be repaired or compensated.
- *Robot self awareness* describes if a robot has built an appropriate model about its state and the environment, or if it is disoriented. In contrast to physical defects, these effects are temporary and can often be resolved quickly without physical intervention by a human.
- *Current activity* reflects which task the robot is performing, and if this task is the best one (within the robot's means) to advance the team's mission. If a robot is idle, the supervisor needs to know the reason, to be able to resolve this situation if possible.
- *Task success potential* addresses the correlation between robot health, robot self awareness and current activity. The supervisor shall be enabled to know if a single robot, in its current status and configuration, is able to solve its task, or if it requires any support or should even abandon this task and continue with another, more appropriate task.

Hence, good SO includes information about past and current states of the robots and the mission as well as predictions about future states. An appropriate knowledge about the current state of robots and mission also requires information about the progress and development in the past, to know how the current state has to be interpreted. Assumptions or extrapolations to future states enable the supervisor to interact with the robots and correct the team's behavior as early as possible. To obtain SO, the information sent from the robots to the supervisor needs to be carefully selected. Details only needed for teleoperation should be omitted, to prevent information overflow. Concrete examples for different applications are given in Section 3.2.

In summary, the supervisor needs to have an overview of the activities within the team and the status of robots and mission. This is a sufficient knowledge base to coordinate a team and to recognize situations that require supervisor interactions. Nevertheless, the required amount of information is limited, and includes mainly qualitative instead of quantitative information, which allows a supervisor to maintain SO for a whole team of robots.

3.2 SO Examples for the Addressed Problem Classes

To get a more concrete idea of SO, here some examples are given for the problem classes of Chapter 2.1.1. Endsley hypothesized that a person's ability to obtain SA is "*a function of an individual's information-processing mechanisms, influenced by innate abilities, experience, and training*" [30]. A study by Chen et al. supported the hypothesis, that a person's spatial ability affects his or her performance of tasks that involve visual scanning [16]. Similarly, the required information to obtain SO is not only dependent on the mission, but also on the supervisor's innate abilities, individual preferences, experience, and the current objectives within the mission. Therefore, only exemplary cases are described, which are not universally valid for all situations.

3.2.1 Urban Search and Rescue - RoboCup Rescue

The examples in this section refer to the USAR problem in the NIST standard test arenas, as described in Chapter 2.1.1.

Mission SO

The supervisor's knowledge about *mission achievement* should include information about mapped and unexplored areas, and the approximate location of detected victims and of hypotheses for potential victims. For the detected victims the supervisor has to know which of them have already been supplied with health kits, and which of them still need support.

For the *mission progress*, besides the knowledge about the capability of the available robots to achieve the pending tasks, the supervisor has to keep track of is the time limit. In a real rescue mission, the chance of survival of trapped victims decreases drastically over time. After the "golden 72 hours" the chance to find victims alive is said to be marginal. At a RoboCup competition, the mission time is limited by the rules.

For *team coordination* the supervisor has to assess if the robots spread properly in space, e.g., if two robots hinder each other because their paths intersect. Furthermore, the supervisor has to be aware of the robots' specific capabilities in relation to the requirements of the individual tasks. For example, if a wheeled robot (which can move fast on flat surfaces) is exploring a very cluttered environment, while a tracked robot (which can climb stairs or rubble piles) is exploring an area with flat floor, this should be noticed (and possibly resolved) by the supervisor. However, for rating the allocation of tasks to robots, not only the capabilities, but also the spatial distribution of tasks and robots has to be taken into account.

Robot SO

Regarding the *robot's health*, the supervisor needs to understand the consequences of the failure of sensor or actuator for the robot's abilities. For example, if the thermal camera fails, this robot cannot search for victims anymore, but is still able to map unknown areas or to bring health kits to detected victims. Similarly, if a manipulator arm breaks, this robot cannot deliver health kits, but can still do all other tasks. In case the flippers of a tracked robot stop working, the robot can still accomplish all kinds of tasks, but can no longer negotiate stairs or very rough terrain, which limits the operational area of this robot.

For *robot self awareness*, the supervisor has to notice relevant differences between the actual state and the robot's internal models of its own status and the environment, that lead to faulty

robot behavior. This can occur if the robot gets stuck because of, e.g., a too low ceiling or cannot negotiate an obstacle, but only notices the stuck situation without detecting the reason. Furthermore, the supervisor needs to know if the robot correctly perceives its own health and the consequences for its abilities.

To understand a robot's *current activity*, the supervisor needs to know the type of the robot's current task (e.g., map an area and/or search for victims), and the location of this task. The location includes the rough position in the map, and environmental features like stairs or obstacles on the way to the target position. Based on this information, the supervisor has to assess if this task is the best possible for this robot to work on. All other open task should be either too far away to be profitable (or not easily accessible because of difficult terrain), be less important (e.g., a victim hypothesis should be verified before an adjacent unexplored area is mapped), or have a low chance of success for this robot because of its abilities.

The *task success potential* shall enable the supervisor to decide if a robot needs human support or should work on another task than the current one. To quickly recognize if the robot has problems in fulfilling a task, it can be helpful to know the total time the robot is already working on this task, compared to the originally estimated time to solve the task. The supervisor has to know if nevertheless the robot can complete the task, how strong the probability of a successful task achievement is affected, or if human support is required for a successful task completion. If, for example, the supervisor notices that a victim hypothesis results from a heater, there is no need for the robot to further explore this hypothesis.

3.2.2 Urban Search and Rescue - DARPA Robotics Challenge

For the DRC, frequently the interactions between a human and the robot exceed the scope of a supervisor. Therefore, the concept of SO is not appropriate as single awareness model for the whole mission. Instead, SO can be seen as a base line, that can be used as an indicator to determine if the robot needs support by an operator. For the phases of closer interactions, a more detailed awareness model like SA is required.

Mission SO

Mission SO is rather simple in the DRC, because only one robot is involved in the mission, and the mission consists of a series of sequential tasks.

To assess the *mission achievement* the supervisor needs to know the status of the current task. For example, for driving the car to the disaster site, the robot has to locate the car, approach the car, find the driver's entry, and enter the car, before it can actually start driving the car. The supervisor needs to know if all tasks, that are preconditions for the subsequent tasks, are accomplished correctly.

Similarly, *mission progress* can be applied straightforward. In addition to knowing the state of execution of the individual tasks, the supervisor should be aware of the elapsed and remaining time of the mission, and have an estimate of the time required to accomplish the remaining open tasks.

Team coordination can be disregarded for the DRC, because there will be only a single robot, not a team of robots.

Robot SO

In the DRC, because of the complexity of the individual tasks it can be expected that any defect can cause the robot to fail entirely in executing a task. Therefore, the *robot's health* includes that all sensors are delivering data. Furthermore, to anticipate defects at the actuators by overloading, the trend of the motor temperature, accelerations, and torques should be monitored.

To assess the *robot self awareness*, the supervisor needs to understand if the robot is fully aware of its current actions and the effects of these actions to the environment. This includes especially the robots capability to judge if a subtask has been fulfilled successfully, or if more work is required to achieve a subgoal of the current task. For example, the robot cannot start driving the car, if the subtask of entering the car has not been achieved properly. If the robot wrongly believes to have entered the car properly, the supervisor needs to notice this discrepancy and intervene in the task execution.

The robot's *current activity* includes, that the supervisor knows the robot's state within the current task, and the robot's activities within this task. For example, in the driving task the supervisor needs to know if the robot is approaching the car, is already sitting in the car, is already driving, or if the robot already reached the destination and only needs to leave the car.

The *task success potential* is based on the robot's current activity and self awareness. The human must understand if the robot within its current autonomy mode is able to fulfill at least the next subgoal of its current task, or if more (or less) human intervention is more promising. To properly rate the task success potential, the human needs more detailed information about the state of the robot and the environment than in the RoboCup Rescue scenario, because each task of the DRC is much more complicated than the whole RoboCup Rescue mission.

3.2.3 Robot Soccer

The examples in this section refer to the problem description given in Chapter 2.1.1.

Mission SO

Mission achievement in a soccer match means that the team wins the match. In the humanoid kid size league, a match ends either after regular time (two halftimes of 10 minutes each), or if the match has a goal difference of 10 goals (i. e., one team scored 10 goals more than the other one). Therefore, for the supervisor it is important to keep track of the scored goals of both teams. If the team is in the lead, the two ways to achieve the mission are either to score even more goals, or to play very defensive and prevent the opposing team from scoring. If the team is behind, the only way to win is to score more goals.

For the *mission progress* the trend over time of the goal difference is important. This can give hints to the supervisor if the performance of the own or opposing team improves or degrades over time, or if the relation stays constant. If the own team is in the lead, the supervisor has to notice if the opponent catches up, because with this information the supervisor can decide if it makes sense to change the current tactic or if it is better to continue with the current strategy.

For an overview about *team coordination* the supervisor must know the current tactic of the team, and which individual roles are implied by this tactic. If the robots coordinate properly, none of the roles may be allocated to more than one robot. For example, exactly one robot at a time should try to handle the ball, and not more than one robot should be a goalkeeper. In general, the robots must not interfere with each other's actions by standing too close to a teammate. Furthermore, in a heterogeneous team, the assigned roles should reflect the robots' abilities. A

robot that is able to pick up the ball with its hands is best suited as a goalkeeper, while a robot that is very agile and good at kicking should preferably play as striker. Therefore, the supervisor needs to know the roles that are allocated to the robots, and each robot's planned actions within its current role.

Robot SO

For the *robot health* of each robot the supervisor needs to know if the camera is working correctly, because otherwise the robot cannot determine its own position and the position of the ball. Defect of the IMU can lead to frequent falls due to missing balancing motions, and can cause false or missing fall detections. This must be noticed by the supervisor. Furthermore, the supervisor needs an overview about the status of the actuators (e.g., motor temperature) and the consequences for the robot's capability to perform the different roles. If the robot can no longer kick the ball without falling, its performance in the striker role can be expected to be very low. However, if the robot can still walk reasonably well and get up after a fall, it can still play as a libero or supporter, where kicking is not required.

Relevant factors of *robot self awareness* for the supervisor involve the robot's internal world model in relation to the real world. The robot needs to know its own position on the field and the position of the ball. The supervisor should notice any major deviations. Also the assumed position of teammates and opponent players should not differ too much from the actual positions. Furthermore, the supervisor has to notice quickly if a robot does not detect a fall, or wrongly assumes a fall although it is still standing or walking.

The robot's *current activity* is described by its current role within the team's tactic. The supervisor needs to understand to which actions this role corresponds, and which is the desired position of the robot, either on the field, or in relation to the ball or other players. If the robot is idle this must be noticed by the supervisor immediately.

The *task success potential* for a soccer role cannot be described as an expected result, because the roles are persistent and cannot be explicitly completed like the tasks in a USAR mission. Instead, the contribution to scoring goals and preventing the opponents from scoring has to be estimated. For the striker the number of shots on goal and goals scored are usually good performance indicators. Similarly, the goalkeeper should ideally catch every shot on goal of the opposing team. If it does not detect all shots, or fails to catch them frequently, the goalkeeper's contribution to the mission goal is very low. These and similar situations have to be noticed by the supervisor, to be enabled to rate each robot's success.

3.2.4 Production Hall Logistics

The examples in this section refer to the problem description given in Chapter 2.1.1.

Mission SO

For *mission achievement* the number of produced and delivered goods is relevant. Therefore, the supervisor needs to know which of the ordered goods have already been delivered and which are still pending. For the pending goods, the supervisor should know the production state, i.e., which products are already in production, and which parts have to be assembled for the complete product.

To overview the *mission progress*, it is important to know if enough raw material is available for all pending goods, and if for every required production step a machine is available and functional.

In case there are any delivery deadlines for some of the products, the supervisor must be able to estimate if these products can be delivered on time.

The team coordinates properly, if they do not block each other's way, and if they do not queue frequently, for example at a storage for picking up material, or at a single machine while other machines are idle. Therefore, to assess *team coordination*, the supervisor needs to know at which storage the robots want to pick up or place material, and at which machines they plan to process the material. The planned trajectories of the robots are important to recognize potential bottlenecks for the robots' paths.

Robot SO

Robot health includes information about the status of each robot's sensors for localization and detection of machines, material, and other robots, the reliability of the actuators used to pick up and deliver material and the correct operation of the actuators used for locomotion. If the components are not redundant within each robot, a defect in one of them implies that the robot must be repaired, because all individual tasks require the same abilities.

Errors in *robot self awareness*, that need to be known to the supervisor, include especially manipulation failures. For example, if a robot does not notice that it failed to pick up material, the supervisor needs to be aware of this discrepancy. Similarly, if a robot does not correctly place a product at a machine or at the delivery zone, the supervisor needs to notice if the robot detects and corrects this error or not.

To understand the robot's *current activity*, the supervisor needs to know which material the robot has to pick up, and where it needs to place it. Furthermore, the current phase of the tasks needs to be known to the supervisor, to understand the current action of the robot. Because of the homogeneity of the robots, only the robot's current location, and not its capabilities, determine if a robot's task is the best possible to advance the mission goal.

To rate the *task success potential*, the supervisor needs to be aware if the robot can perform all required actions without failures. If the robot is working on a product that has a delivery deadline, the supervisor needs to judge if this deadline can be met, and therefore needs an estimation of the remaining time to complete the task.

3.3 SO in Contrast to other Awareness Concepts

Human Team Hierarchies

The concept of SO follows the same idea as the communication in human team hierarchies (potentially involving robots at the lowest hierarchy level). For example, a search team manager at a rescue site is informed by task force leaders about important events, e.g., if a void has been searched and the team is available again. This concept is "*simple, reliable, and reduces information overloading and distractions to decision-makers*" [84]. In general, in human teams a common approach is to have a team leader, who monitors the progress and intervenes if the progress is not as desired [34]. In contrast to the concept provided here, the team leaders obtain the information from other humans, i.e. team members or leaders of smaller teams. However, if instead a team of robots shall inform a human leader about their progress to support the leader's SO, they cannot judge a situation in the same way as a human. Therefore, SO gives a guideline about which information the robots have to send to the supervisor and when. A method that enables the robots to obtain this information is presented in the next chapter.

Workspace Awareness

The workspace awareness elements defined by Gutwin et al. [53] (c.f. Chapter 1.2) correlate in many points with the topics defined for mission SO and robot SO. For example, the current activities of the team members and the effects of their actions are present in both, workspace awareness and SO. However, the focus of workspace awareness is on coworkers, who have to be aware of the changes in their direct surrounding, but do not have to maintain an overview of the whole project. This corresponds more to the awareness each robot should have about its teammates to enable cooperations. But still the individual points of workspace awareness cover a large part of SO, because also the supervisor must be informed about the actions and interactions of the robots. However, because of the different focus of workspace awareness, important aspects of SO are not included, like mission progress, team coordination, and task success potential. Also, aspects about the status of each individual, like health and self awareness, are not present in the definition of workspace awareness. Overall, workspace awareness is in many points similar to SO, as it also requires broad information without specific details, but lacks some important coordinative features because it addresses peers instead of supervisors.

HRI Awareness Framework

The HRI awareness framework by Drury et al. defines five awareness types [26] (c.f. Chapter 2.2.2), of which two are relevant to be contrasted with SO (the other three describe awareness between human collaborators, between the robots in the team, and the model the robots have of the human). Human-robot awareness includes basically the same information as SA, which will be addressed later in this section. Humans' overall mission awareness refers to the team's progress towards mission achievement. This is also covered by mission SO as part of the complete SO definition (mission achievement and mission progress). However, additionally the team coordination is included in mission SO, which is not explicitly address by the HRI awareness framework.

Situation Awareness

SA has its origins in civil and military aircrafts, e.g., pilots in air combats, and has been applied to many other situations that focus on decision making in general, e.g., air traffic control and strategic systems [30]. In the last years, it has also been applied to HRI for the interaction role of an operator, especially in the domain of USAR [110]. SA focuses on closer relations between the human operator and the (partially autonomous) system, while SO is designed for human supervision of almost fully autonomous robot teams. Particularly, SO addresses team aspects, like overall mission progress and team coordination, which are not covered by an aggregation of individual robot SA.

Because of the different requirements of operators and supervisors, SA includes much more detailed information than SO. For example, an operator of a rescue robot who has obtained SA is supposed to know the robot's exact position, orientation, and relation to the environment, like the distance to surrounding obstacles and maneuverability on the current soil. For SO in contrast, the supervisor only needs to know an approximate location, e.g. in which room the robot is currently located, and if any defects or environmental conditions hinder the robot in fulfilling its task. This reflects the *task success potential* as a part of robot SO.

In general, SA consist of three levels: perception, understanding, and projection. It would overburden a supervisor, if those three levels would have to be achieved for each robot in the team individually. Instead, the robots can take over level 1, and directly provide the supervisor with information corresponding to level 2 and 3. Because the robots are autonomous, this information

exists anyway, it remains to make the right parts of the existent information available to the supervisor. In that way, the details that are not relevant for SO can be dropped already by the robot, and therefore the supervisor's cognitive load is reduced. This allows the supervisor to concentrate on other tasks, that require the specific capabilities of a human, instead of doing tasks that can easily be done by the robots, as described in Chapter 1.1.

Besides the reduced cognitive load of the supervisor, the reduced communication overhead is a main advantage of SO over SA. Less details are needed, therefore less data have to be communicated to obtain SO compared to SA. Furthermore, the information required for SO changes rather slowly, hence no continuous communication channel is required, because SO can be obtained by discrete information from the robots, instead of, e.g., continuous video streams. This is an important factor in real-world applications where the available communication bandwidth is usually limited. A method to obtain SO, that makes use of this fact, is presented in Chapter 4.

However, because SO is, like SA, based on the perception and projection of information, many research results from SA can also be transferred to SO. The following results from [30] also apply to SO:

- It cannot be defined in general, for different applications and robots, which information the human needs to receive to achieve SA/SO.
- SA/SO is influenced by several human factors, like attention, working memory, workload and stress.
- The possibility to obtain SA/SO is dependent on which information the system provides, and how this information is presented.
- Trained or experienced users can achieve a high SA/SO easier than novices. Furthermore, some people are in general better in obtaining SA/SO than others.
- Achieving SA/SO is a process over time, not a discrete action.
- Good SA/SO can increase system performance, but bad performance does not necessarily indicate bad SA/SO.

Endsley concludes, that there are two factors in a system, that influence how good a human can obtain SA: 1) which information does the system provide, and 2) how the available information is presented to the human. This also applies to SO. The first factor is addressed by the communication concept presented in Chapter 4, while for the second factor some design principles are summarized in Chapter 6.



4 Communication Concept for Obtaining Situation Overview

The SO examples in the previous chapter show, that continuous information can be transformed into discrete events, which altogether compose the information required for the supervisor to obtain SO. Those events are called *SO events* in the following.

The communication concept is inspired by human teamwork (c.f. Chapter 1.2), where team leaders are informed about the general progress and potential problems, but are not provided with all details about the execution of a task.

The three major steps of the communication concept are

- detection of relevant events,
- classification to different message classes, and
- adjustment of the amount of sent messages using policies.

This concept has several advantages compared to continuous data streams without considering any semantics. Besides the general reduction of communicated data, selective observation of special happenings is made possible, regardless of the source of information. Hence, the presented communication concept supports the supervisor in obtaining situation overview, while simultaneously decreasing the communication overhead, compared to standard teleoperation interfaces.

4.1 Event-based Communication

The central elements for the presented communication concept between a robot team and a human supervisor are *events*. Following the definition of Hinze et al. [57], an event is the observation of “*a significant change in the state of the universe*”. Two identical observations of the universe, that only differ in time, describe two different events, because time is an integral part of the universe. Hence both, changes of a state as well as the fact that a state does not change over time, can be modeled as events.

4.1.1 Background: Complex Event Processing

The research field of Complex Event Processing (CEP) deals with the question of how to detect and handle events in communication systems, for example in databases or wireless sensor networks (WSNs). An overview about CEP and its application to different fields is provided in [57]. In the following, a comparison is given between CEP for robot teams and the exemplary application of WSNs.

In WSNs, several (up to hundreds) of distributed sensor nodes are used to detect events, e.g., human presence or fire. Simpler events are combined to detect complex events, that are aggregations or patterns of several events. The analogy between CEP as used in WSNs and robotics is, that there are several sensors and pre-processed data available, based on this information, certain events or states of the robot or the world have to be detected. The key differences are, that a robot has less, but more reliable sensors than in a WSN, the sensors are more complex and deliver not only scalar values. Furthermore, the “network” is static, apart from sensor failure, because

a robot's sensors are not entirely distributed, but are all physically connected. Therefore issues like time synchronization and timeliness can be disregarded for CEP a single robot. In case also events have to be detected that involve more than one robot, also the aspects of synchronization and timeliness have to be considered. However, in robot teams this is usually handled by the robots' middleware.

The tasks and capabilities of a robot team are fundamentally different from those of a WSN: robots can interact with the environment in time and space, while a WSN can only monitor the state of the environment over time. Hence, the robots can base expectations about changes in the environment on their own actions. Furthermore, the robots' mobility allows to systematically collect data at locations where a high information gain is estimated.

Event-Condition-Action Rules

In the area of CEP, *event-condition-action (ECA)* rules are used to describe the desired system behavior based on the detected events. In the context of this work, the *condition* is described by policies (c.f. Section 4.3), and the *action* is simply to send the event to the supervisor, if the condition is met.

However, although the primary purpose of the events in this context is the communication with the supervisor, also an event-based behavior of the robot control software can be realized, using other ECA rules based on the detected events.

Event Algebras

The events that shall be detected by the robots can be very diverse to many aspects. Some are just special variables exceeding thresholds, others are regular patterns that have to be detected, or several occurrences of different events simultaneously. The detection of every single event could be programmed manually, but this is very time consuming, can lead to many failures, and usually duplicates lots of code.

To overcome this, CEP incorporates the concept of event algebras, which provide operators to combine several events into an aggregated (or complex) event. In this context, *simple events* are discrete events, that can be directly detected without aggregating more information, e.g., the change of a status, or incoming sensor data. *Complex events* are events that are compositions of (simple or complex) events, or events enhanced with external information. These compositions can be, for example, two events occurring simultaneously, an event chain, or patterns.

Examples for event algebras are HiPAC [20], SNOOP [13], REACH [9], ADAM [24], SAMOS [42]. Those algebras provide operators as conjunction, disjunction, or sequence, to combine two or more events to a complex event. The algebras vary in complexity and versatility. Depending on the application, an appropriate algebra needs to be chosen, that satisfies all needs, but is not too complex, hence being more difficult to understand and leading to higher implementation efforts.

4.1.2 Definition of Events

For the use in this thesis, an event consist of the respective type, the detection time, the name of the robot that detected the event, and a list of tags, which will be defined in Section 4.2. Additionally, an event can optionally be enhanced with other relevant information, called event payload. The payload can consist of several of the types described in the following.

Location

If an event corresponds to a specific location, the coordinates of this location can be attached to the event. This can be used, for example, to describe the location of a detected object of interest, or to mark a location where a robot had difficulties in negotiating obstacles.

Image

Images can be attached to an event to visually support a statement. For example, if an object of interest was found by a robot, an image of this object enables the supervisor to decide if it is a true or false positive. The images can be either exactly like they were taken by an on-board camera, or can be enhanced with further information, e. g., perceptions from an object detection algorithms can be highlighted.

Map

In cases where the location is important, this can be further supported by a map (the complete map or a specific region of interest) learned by the robots. This map shows on the one hand the area explored by the robots, but can also be further enhanced by, e. g., the robot's current location, the location of objects of interest, the path traveled by the robot, or the current goal of the robot.

Other Events

If an event is composed of other events, i. e., is a complex event, these events can also be attached to the new event, so that it gets clear how the event is composed.

Text

Finally, an event can be accompanied by an arbitrary parametrized text, that describes details of the event. For example, if an object of interest has been detected, and an image of this object is sent to the supervisor, a text can state how certain the robot is about the detection.

4.1.3 Event operators

The operators that are used within this thesis either act on input data like single variables for generating simple events, or connect two or more events to a complex event. In addition to events that are directly derived from the system (e. g., detections in an image), the following operators are used for the below examples:

- *Variable change:* An input variable is compared to an initial value (e. g., the value of this variable at a certain point in time, or a predefined threshold). An event E is raised as soon as the variable exceeds these predefined bounds. This operator can be applied to all types of variables that allow comparisons, e. g., discrete or continuous numbers, but also multi-dimensional vectors. In case the variables can be ordered, E can be enhanced with information if the value increased or decreased.
- *Negation:* An event $!E$ is raised, if E does not occur within a predefined timespan.
- *And:* An event E_3 is raised, whenever E_1 and E_2 both occur within a predefined timespan in arbitrary order.

- *Or*: An event E_3 is raised whenever E_1 or E_2 occur.
- *Sequence*: An event E_3 , described as the sequence E_1, E_2 , is raised when E_2 occurs after E_1 in a predefined timespan. Longer sequences (E_1, E_2, E_3, \dots) can be realized as sequences of sequences. E_4 can be defined as interrupting event for a sequence, i. e., E_3 is only raised if E_2 occurs within the required timespan after E_1 , before E_4 occurs.
- *Count*: This operator counts the occurrence of events of the type E_1 within a given timespan, and raises an event E_2 either every time a new event E_1 arrives, enhanced with the number of occurrences of E_1 within the timespan, or raises E_2 as soon as E_1 occurred at least n times.
- *Collection*: A collection is similar to the *and*-operator, but allows to include several distinct events of the same type. A collection can also be expressed by a combination of *and* and *count*, but is used as separate operator for brevity.

4.1.4 Event Examples for Reference Classes

Besides general events, that are applicable to arbitrary robots and scenarios (e. g., battery status) there are also many events closely related to the considered problem. In the following, some examples for the reference problems of Chapter 2.1 are presented.

Urban Search and Rescue – RoboCup Rescue

In the USAR scenario, events are a useful tool for reducing the amount of communicated data. For example, it is not necessary to continuously transmit every camera image to the supervisor. Instead, an event E_1 is raised if the robot detects human evidence in a region. The event is enhanced with a camera image of the region of interest as a payload.

Similarly, the map does not need to be sent periodically, e. g., in situations when the robot does not move. Instead, the position of the robot is tracked, and an event E_2 is raised, whenever the robot's position changed for more than a given threshold, compared to the location of the last event of type E_2 . The map, the robot's traveled path and current position are attached as payload to E_2 .

The robot periodically raises an event E_3 while it wants to move, with the desired velocity as payload, or an event E_4 while it does not want to move. Event E_5 , the *sequence* $E_3, !E_2$, indicates that the robot is stuck at a location, although it plans to move. The timespan for detecting the sequence must be dependent on the velocity given in E_3 . The sequence detection is interrupted as soon as an event of type E_4 is raised.

Urban Search and Rescue – DARPA Robotics Challenge

For the DRC, consider the task of removing debris blocking the entry of a building. Because the weight of the rock is not known in advance, it is not clear how much force the robot has to apply to be able to lift the rock. If E_1 describes an increase of the applied force by a certain amount, with a payload denoting the applied force, and E_2 describes a significant change of the joint angles used for lifting, E_3 , given as the *sequence* E_1, E_2 is raised as soon as the force is high enough for lifting the rock. The event E_4 , given as the *sequence* $E_1, !E_2$ is raised if the force in E_1 is not high enough.

For the example of climbing a ladder, events $E_5 - E_8$ describe if the robot's hands and feet are placed safely on the ladder and are raised periodically as long as this state does not change. If at least three of the four contacts are reliable, the robot can be considered to be stable on the ladder. This can be modeled as a complex event of the form: $(E_5 \text{ and } E_6 \text{ and } E_7) \text{ or } (E_5 \text{ and } E_6 \text{ and } E_8) \text{ or } (E_5 \text{ and } E_7 \text{ and } E_8) \text{ or } (E_6 \text{ and } E_7 \text{ and } E_8)$.

Robot Soccer

The first set of events in the soccer application is related to the trend of a match. These events can be derived from the number of goals scored by the own team G_1 , and the number of goals scored by the opposing team G_2 . These input data are provided by a referee box during a match. Event E_1 is raised every time the own team scores, i. e., there is a *raise of the value* of G_1 . Likewise, E_2 is defined as scoring of the opposing team, i. e., there is a *raise of the value* of G_2 . Based on this, a run of the own team can be defined as a *sequence* of E_1, E_1, E_1 , without E_2 occurring in between. A run of the opposing team is defined analogously. The trend of a match, e. g., while playing a specific tactic, is given by the ratio between occurrences of E_1 and E_2 , which can be measured using the *count* operator.

The second example for soccer is related to the health of a single robot, in particular its kicking abilities. A simple event E_{5j} is raised whenever the robot performs a kick of type j . If the robot falls over (during a kick or in any other situation), an event E_6 is raised. If a kick is stable, i. e., the *sequence* $E_{5j}, !E_6$ is detected, an event E_{7j} is raised, otherwise, if the *sequence* E_{5j}, E_6 occurs, E_{8j} indicates an instable kick. Of course, a single kick cannot give significant information about the general stability of a specific kick motion. Instead, the *count* operator can be used for counting the events E_{7j} and E_{8j} in a certain timespan, e. g., 3 minutes. The ratio between these two values can be used as a stability indicator for kick type j .

Production Hall Logistics

In the production hall logistics example, events can be used to monitor the production status of the different products. For example, the placement of some material a at a machine b generates an event E_1 . The complete *collection* of all required material is a new event E_2 , denoting that machine b is ready to start the production. The collection for E_2 can also involve several distinct events of the same type, for example, to require 4 legs (event E_3) and a seat (event E_4) for being able to produce a chair, E_2 is defined as the collection $\{E_3, E_3, E_3, E_3, E_4\}$.

4.2 Event Classification and Tagging

The events described in the previous section provide a good basis for a human supervisor to obtain SO. However, if the supervisor gets simply flooded with messages, critical messages may get lost between lots of status information. To allow an interface to sort the events and to highlight important messages, the events are tagged: on the one hand according to their criticality, and on the other hand based on topics.

The classification of the messages brings several advantages to the usability of the event-based communication. On the one hand, it allows to filter the events, and only send messages of a specific criticality or belonging to a specific topic, as will be shown later. On the other hand, it is possible to sort the events by category and to use different representations of messages of different criticality level at the interface. For example, warnings and errors can be marked using different colors or pop-ups at a graphical user interface, or can be further emphasized using sound.

4.2.1 Criticality Level

To obtain SO, regular events that correspond to planned changes of a robot's state or the environment are equally important as unexpected happenings or critical incidents. However, events of different criticality need to be handled differently, and therefore, before being sent to the supervisor, the events get assigned a criticality level.

Usually, logging systems for software development (e. g., log4j (<http://logging.apache.org/log4j>), log4net (<http://logging.apache.org/log4net/>), or NLog (<http://nlog-project.org/>)) use five stages for messages: debug, information, warning, error, and fatal. The concept provided here is designed to support a human supervisor, who is not familiar with the implementation details of the robot software, therefore the debug-level can be omitted, because these notifications would not advance the knowledge of the supervisor, but rather lead to confusions because of too many technical details. Fatal are usually those errors, that cannot be handled properly and lead to program termination. Thus, these notifications cannot be communicated. Therefore also the fatal-level is omitted here.

In summary, there remain three notification levels, to be used by the robots:

- *Information*: Events, that occur as expected, but can help the operator to obtain SO, e. g., an agent has accomplished a subtask of the mission, or a regular notification about the battery status.
- *Warning*: Events, that do not severely disturb the overall functionality of an agent, but indicate possible malfunctions or problems that can arise in the future, e. g., if the execution of a task takes much longer than expected, or if a robot's battery charging level is low.
- *Error*: Urgent events that indicate serious failures, for example failure of a sensor or of an actuator.

Although the fatal-level is not used here, the supervisor still needs to be aware of the possibility of a complete system failure. To support this, *sign of life* events are used, that send minimalistic events at a predefined frequency. A user interface can raise a warning in case the event fails to appear for a long time, which indicates that the robot does not communicate with the interface anymore. This can be either due to a complete system failure, or simply because the robot left the communication range or was manually switched off. The exact reason cannot be determined by this method, but at least the supervisor is informed about the absence of a robot.

As concrete examples, notifications of *information*-level are used to send reports about a robot's battery status, to inform the supervisor about start and termination of subtasks, or successful checks of the status of a sensor. *Warnings* are sent, e. g., if the battery charging level drops below a certain threshold, the execution of a task takes much longer than expected, or if motion commands do not result in the expected movements. *Errors* are sent, if a sensor or actuator fails to work, or if the battery charging level drops below a critical value.

4.2.2 Topics

Semantic tags, or topics, can be used to reflect the different tasks the robots are working on, functional or mechanical components of the robots, or more high-level topics like a robot's status or goals. In general, tagging allows to map events to the different categories of SO (c. f. Chapter 3.1),

but also to arbitrary user-defined topics. Because the tags can be related to several parts of the system or overlapping topics, one tag per event is not sufficient. Therefore, each event can have a list of tags.

As an example, consider the three event types E_1 : `victim.found`, E_2 : `victim.seeEvidence`, and E_3 : `victim.exploreHypothesis` of a search and rescue mission. All three can be subsumed under the general tag `victim`. However, E_2 can also be tagged to `perception`, while E_1 and E_3 define the beginning and the end of a task, and therefore belong to `task execution` or `high-level behavior`. Other events can, e.g., be related to simultaneous localization and mapping (SLAM), or to the general progress of the search.

As another more general example, consider a robot that sends a `signOfLife` event every ten seconds, a `battery.level` event every time the charging level decreases by 10 %, and a `sensor.failure` event whenever a sensor stops working. All three can be grouped as `robot.status`, which allows a supervisor to specifically observe all events related to a robot's health.

To guarantee a high flexibility, the tags can be adapted during runtime. This allows the supervisor to define new categories on the fly, add event types to existing categories, or remove event types from single categories.

In addition to the manual tagging, some tags can also be generated and mapped to the events automatically using name prefixes. For example, events related to victim detection in a USAR mission all have names of the form `victim.*` (e.g., `victim.found`, `victim.newHypothesis`, `victim.discarded`, etc.).

The mapping of events to tags is stored in a configuration file, to prevent repeated redefinition of the same mappings on every system restart. Only manually defined tags and mappings have to be stored, because the automatically generated tags can be reconstructed easily at every system start.

4.3 Control of the Event Flow

Providing information about every single part of a system can quickly lead to information overload and increased workload for the human [93]. After providing tags for sorting the events, the second step against information overload is to communicate only those events, that are actually of interest to the supervisor. Depending on the current focus, the supervisor does not require all available events of the system for obtaining SO. Policies can be used for defining which events are required. A policy manager collects all policies and communicates exactly the requested messages to the supervisor. In that way, it is possible to reduce the quantity of the communicated data, while simultaneously increasing the quality of information for the supervisor.

4.3.1 Policies

The amount of messages that are sent to the supervisor needs to be controlled carefully. On the one hand, too many messages can result in information overflow and supervisor stress. On the other hand, too few messages lead to a loss of situation overview. In general, there should not be any static rules about which events shall be communicated to the supervisor, and which decisions the robot should take autonomously or with some support by the supervisor. Rather, this is highly dependent on the current mission, the supervisor's preferences, and the supervisor's trust in the system.

Events	Tags	T_1	T_2	Sequential policies for tags	
				1. $S+ \rightarrow T_1$	2. $S- \rightarrow T_2$
E_1		x	-	$S+$	$S+$
E_2		-	x	D	$S-$
E_3		x	x	$S+$	$S-*$
E_4		-	-	D	D

Table 4.1: Resulting event policies after sequential definition of tag policies. First an $S+$ policy is defined for tag T_1 , afterwards an $S-$ policy is defined for tag T_2 . The rightmost column shows the resulting policies for the different event types at these two timesteps. The resulting conflict is marked with $*$.

Policies can be used to define the bounds of an agent’s autonomy [8, 61]. In the scope of this thesis, two policy types are sufficient: an $S+$ policy requires an agent to send a message, while an $S-$ policy is a request not to send a message.

This concept allows to dynamically regulate which agent should send what messages, so that it can be adapted to the supervisor’s preferences and current focus. As an example in the USAR scenario, a supervisor without trust in the robots’ autonomous victim detection might want to get informed every time human-like temperature is detected with a thermal sensor, while a supervisor with more trust might be satisfied getting just the hypotheses that are positively verified by the robots.

Defining policies for every possible message would be highly inefficient. By means of the tagged events defined in Section 4.2, policies can be defined for groups of messages, according to their criticality, or according to a semantic topic. Sets of policies can be stored and loaded, dependent on the current mission, or even situation dependent. Further policies can be defined by the supervisor.

4.3.2 Policy Manager

A policy manager collects all active policies, determines the resulting policy for each event, and resolves potential conflicts of contradicting policies.

Each event can have one of three different policy states: $S+$, $S-$, or D (default). If no policy is defined, all states are set to D . The default value can be defined centrally, and allows the supervisor to decide if the system behaves generally communicative or silent.

If a policy P is defined for a tag, the status of all events that are mapped to this property is set to P . Conflicting policies are resolved either by heuristics, or manually by the supervisor. If the old status of an event is D , no conflict occurs, and the status is simply overwritten. In that way, the system behavior always complies with the most recent policies. In case an event already has a policy $S+$ or $S-$, which is different to the new policy P , the status of this event is marked as conflicting.

Table 4.1 shows an example for resulting event policies after defining policies for some tags. An “x” in the table indicates the mapping of an event to a tag. Initially, all event policies are set to the default value D . Two policies are defined sequentially. First, a policy $S+$ is defined for tag T_1 . Because events E_1 and E_3 are mapped to T_1 , also the corresponding event policies are set to $S+$. Second, a policy $S-$ is defined for tag T_2 , which is a tag for events E_2 and E_3 . Hence, also the event policy of E_2 is set to $S-$. For event E_3 a conflict occurs, because this event policy

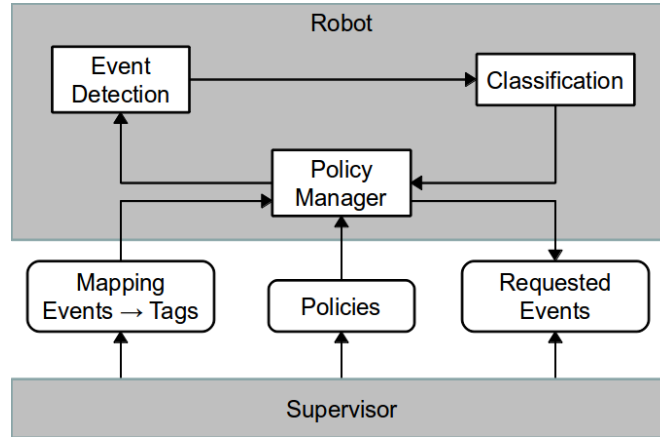


Figure 4.1: Visualization of the interactions among the different components of the event-based communication concept.

has been set to $S+$ because of T_1 , and is now overwritten because of T_2 . As described above, the event policy is preliminarily set to the most recent policy ($S-$), but is marked as a conflict (indicated by a $*$ in the table) that has to be resolved.

Three different modes can be used for solving a conflict. In the first mode, the last policy always overwrites older ones, hence the conflict is simply ignored. In the second mode, the policy for an event in conflicting state is set to the default value D , and hence complies with the generally desired communicativeness. The third mode requires the human supervisor to resolve the conflict using queries (c. f. Chapter 5.1).

4.4 Discussion Communication Concept

The developed communication concept combines well established methods from different fields, to enable a human supervisor to obtain situation overview on a high level. This method is inspired by loosely coupled teamwork among humans. CEP gives the possibility to easily detect events that would require lots of efforts when programmed by hand. The semantic tagging and classification according to a criticality level allows to dynamically regulate the amount of messages sent to the supervisor using policies. This allows to specifically send data to the supervisor, that can be expected to have a high gain of information, while omitting data that does not advance the supervisor's SO. In that way, also the use of network capacity can be controlled, if low bandwidth is an issue.

Overall, the three components are connected in a loop with external feedback from the supervisor, as shown in Figure 4.1: CEP detects important events, which are then classified using criticality levels and semantic tags. The policy manager then decides which of those messages are sent to the supervisor. For closing the loop, the policies can be adapted during runtime, and therefore it changes dynamically, which events have to be detected by the CEP system.

In summary, events are used as a central communication element between the robot team and the human supervisor to coordinate mission achievement. Events are adequate to provide the supervisor with extensive information about regular ongoings, unexpected happenings, and critical errors of the robot team, and therefore are a good basis to obtain SO. The human is given a knowledge base to keep track of the advancement of the mission, to support the robots if they experience any problems, and to coordinate the robot team if necessary.



5 Interactions between Supervisor and Robot Team

Having obtained SO, the supervisor is able to interact with the robot team on a high level. However, the supervisor cannot act on a robot's state space (i.e., teleoperate a robot), because this would require a complete SA instead of SO.

Interactions with the human can be initiated by the robots by transferring decisions as queries to the supervisor. The robots' level of autonomy can be adjusted by adapting the amount of queries that have to be answered with human support. This query concept is presented in the first part of this chapter.

The human-initiated interactions are separated into commands that modify the mission of the robots and commands that affect the allocation of tasks to robots. For instantaneous task assignment, this can be achieved by only modifying the input data for the task allocation algorithm, hence, different algorithms can be exchanged transparently. For time-extended task assignment, the commands are translated into soft constraints or hard constraints for a mixed-integer linear program. Both approaches are presented in this chapter.

The focus of this concept is on the messages that are sent from the supervisor to the robots, and how these messages are interpreted. In particular, the specific interface design is currently disregarded.

5.1 Robot-initiated Queries

In human-robot teams, it is often desirable to adjust the robots' level of autonomy (LOA). The required LOA is dependent on different factors, like the mission in general, the current state of the robots and the environment, the supervisor's trust in the robots' autonomy, and the supervisor's personal preferences. Within the developed concept, the supervisor can use policies (c.f. Chapter 4.3) to define the borders of the robots' authority. The robots can initiate interactions with the supervisor if fully autonomous execution of a subtask would violate their granted authority. The event-based communication concept described in Chapter 4 is used as a basis to enable adjusting the LOA. Queries are used to transfer high-level decisions from the robots to the supervisor.

5.1.1 Queries

Queries are a special form of events as defined in Chapter 4.1, that are used to transfer a decision from a robot to the supervisor. They enable the robots to get decision support from the supervisor in situations where either the robots do not have sufficient information available, or are not granted enough authority for taking a decision autonomously. Queries comprise all properties of regular events. In particular, queries can be tagged to different topics, and can have payloads like text or images. Therefore, the amount of queries, and hence the LOA, can be regulated with the same policies that are used for the regular events. As additional payload, a query contains a description of the required decision and a set of possible solutions. This can be either a fixed set of static solutions, or can allow free inputs if appropriate, for example for numerical values. For each query, a response event is expected, containing one of the originally stated possible solutions.

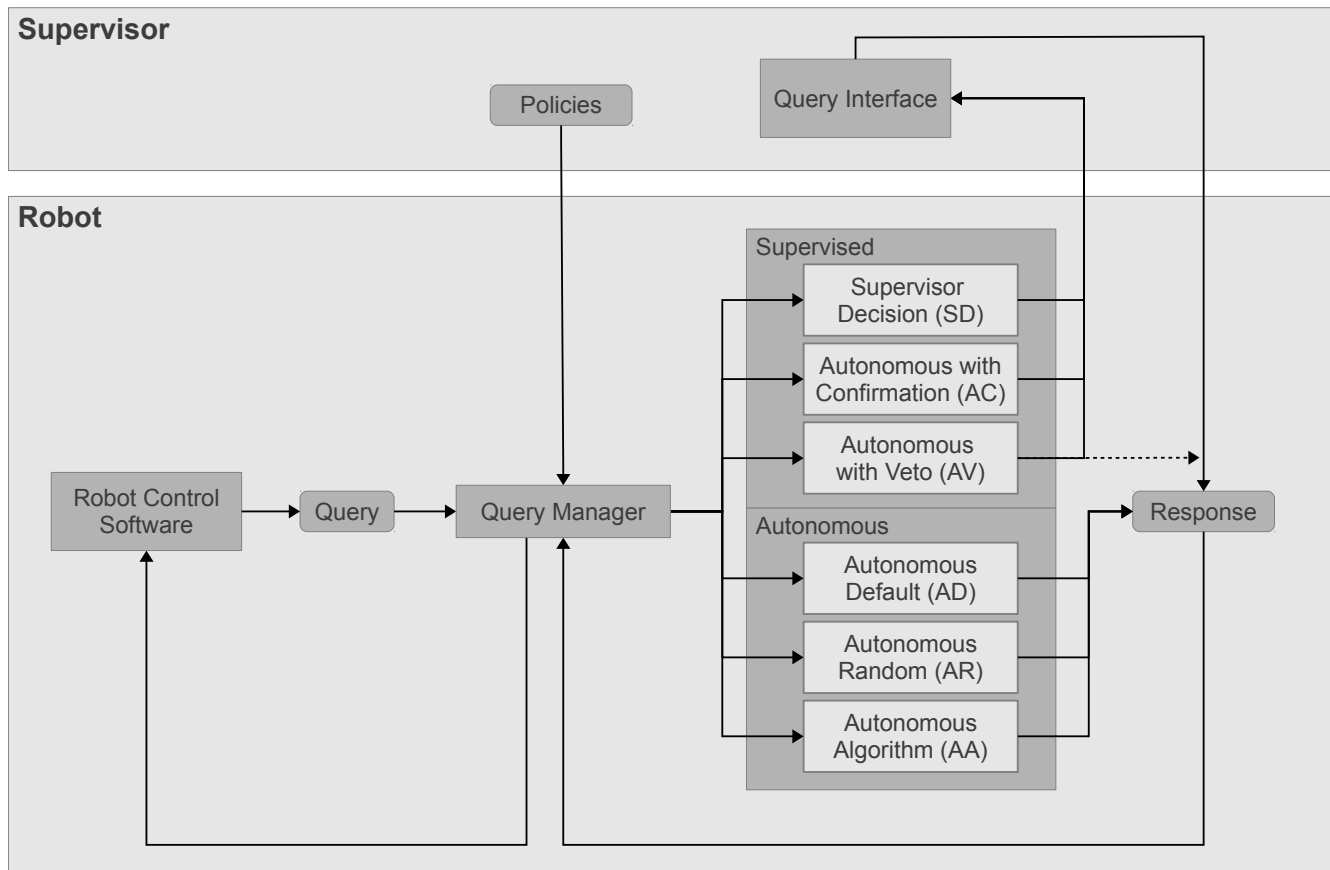


Figure 5.1: The query manager enables the supervisor to dynamically adapt the robots' LOA by selecting different query modes.

It can take several seconds or minutes after sending a query, until an answer is received. Therefore, a robot that sends a query has to either stop its current activity and wait for the response, or needs to be able to respect all possible answers, even after the state of the robot and the world have changed. Hence, queries should be used with care to ensure an autonomous behavior without too many interruptions. However, full autonomy has to be traded off for the amount of queries and resulting interruptions for each scenario independently.

After answering a query, the supervisor needs to know if the response has been received, and if the actions are executed properly, to stay in the loop [88]. This is handled using notification events (c. f. Chapter 4) and belongs to the supervisor's SO. These events correspond to a nodding of a human to signal that the command has been received, which is crucial for communication among humans [89].

5.1.2 Query Manager

To allow adapting the LOA during runtime, a query manager is used, similar to the policy manager in Chapter 4.3.2. It collects all queries that are generated by the robot control software, and determines the response, taking into account the currently granted authority of the robot.

In case a decision is transferred to the supervisor, there are three possible modes for presenting the queries:

-
- *Supervisor decision (SD)*: All possible answers, including variable parameters, are presented, without differentiating between the options. The supervisor has to select one of the options, and if necessary specify variable parameters.
 - *Autonomous with confirmation (AC)*: In addition to SD, a potentially best solution is determined by the query manager, and highlighted for the supervisor. The human can either confirm the preselected solution, or select one of the other available options.
 - *Autonomous with veto (AV)*: The possible solutions are presented in the same way as with AC, but if the supervisor does not veto the suggested solution within a specific time t_{veto} , this answer is given automatically. Before this time is over, the supervisor can veto and select another option for execution, or adapt the parameters for the selected solution.

If a decision is not transferred to the supervisor, the robots have to select a solution autonomously. Also in this case, several different modes are possible:

- *Autonomous Default (AD)*: One of the possible solutions is defined as default answer, which is always selected.
- *Autonomous Random (AR)*: One of the possible solutions is chosen randomly.
- *Autonomous Algorithm (AA)*: An algorithm is executed, that determines the best solution based on the current status of the robots and the environment.

Figure 5.1 provides an overview about the available query modes and the functionality of the query manager. With the different query modes, it is possible to model lots of decisions in the robot control software as queries. The supervisor can use policies to define the mode for each query type, or for groups based on tags, analogous to the policies for regular events. This allows to transparently switch between autonomous and supervised decisions, because the querist always uses the interface to the query manager. Most of the queries should be answered using one of the fully autonomous modes (AD, AR, or AA), to avoid long periods of idle time for the querist. The more queries have to be answered with supervisor support, the lower is the robots' LOA, and the higher is the workload of the human supervisor.

When selecting the mode for the different query types, the supervisor should not be bothered with queries that can be easily answered autonomously, to avoid effects of human fatigue and complacency [92].

The different query modes even allow to implement algorithms to learn on-line from human decisions. To achieve this, a query type should start in SD or AC mode. The decisions, together with the current state of the robot and the environment, can be used as input data for a learning algorithm. Having obtained enough samples, the mode can be switched to AV as a test phase. The number of vetoes can be seen as an indicator of the algorithms prediction capability. As soon as the supervisor's trust in the algorithm is high enough, the mode for this query type can be set to AA, with the newly learned algorithm as decision maker.

5.1.3 Examples for the Reference Problems

In this section, some examples are given for queries in the reference problems of Chapter 2.1.1.

Urban Search and Rescue - RoboCup Rescue:

Whenever a robot finds a potential victim, the rules require the robot to stop close to the victim, and wait for a confirmation by the supervisor, before the victim is scored and the robot is allowed to continue with the mission. Therefore, the *victim verification* query has to be answered in SD or AC mode.

A second application of queries in this context is *object recognition*. Humans are usually much better in visual object recognition than robots (c. f. Chapter 1.1). Therefore, detections of objects of interest (OOI) can be send as AV queries to the supervisor. If, for example, the human identifies a false positive victim (e. g., a radiator that caused the processing of the thermal images to generate a victim hypothesis), the corresponding model and the respective victim verification task can be deleted, and therefore the robot does not have to further investigate this hypothesis.

Urban Search and Rescue - DARPA Robotics Challenge:

In the DRC, a lower LOA can be appropriate, to reach a high reliability. The high level behavior is modeled as a hierarchical finite state machine, which enables to send queries for important state changes.

One application for queries triggered by state changes is, to *select the autonomy mode* of the next state. Each subtask, modeled as a state, can be executed with different autonomy modes, ranging from fully autonomous execution to complete teleoperation. As soon as a state with different available autonomy modes is entered, a query is generated to select the appropriate mode. Depending on the desired LOA, this decision can be fixed before the start of the mission (and hence the query mode is AD), or is transferred to the supervisor as SD or AC query.

A second application for queries within a state machine is a *state transition monitor*. A query is formulated to determine if the conditions for leaving the current state and entering the next state are completely fulfilled. For example, if the robot has to grab a tool, it needs to be confirmed that the grabbing was successful, before the robot can continue working with this tool. The query mode is dependent on the desired LOA and on the availability of an appropriate algorithm for checking the grabbing state.

Robot Soccer:

In a robot soccer match, human interactions are only allowed during interruptions of the match, therefore also queries cannot be applied extensively. However, a good application is an *autonomous tactics change*. If the robots detect that the current tactic is not promising enough, they can try if a different tactic increases the team's performance. As a query, this tactics change is either send in AV mode or in AC mode, depending on the supervisor's trust in the algorithm that rates the tactic's performance and suggests the tactics change.

Similarly, each robot can monitor the performance of its current role. If, e. g., the goalkeeper detects that it is either not needed (because the opponents do not shoot on the goal) or is not beneficial (because it cannot block the opponent's goal shots), the robot could instead act as an additional field player, to potentially contribute more to the team's success in that position. Also in this case, a tactics change can be initiated as a query.

Production Hall Logistics:

In case two product types A and B require the same material M, but not enough material is available to produce both A and B, the decision about *which of the products to prefer* can be formulated as a query. This decision should be taken in SD or AC mode, because a human can

usually rate the importance of A and B based on implicit knowledge, that is not available for the robots.

5.2 Human-initiated Instructions

Based on SO, the supervisor can detect situations where human intervention can be helpful for increasing the team's performance. This is enabled using commands for coordinating the robots' actions and for refining the mission. Some types of general application independent high-level commands are presented in the following, afterwards some concrete examples are given for the reference problems.

5.2.1 Application-independent Supervisor Commands

Following the definition of a supervisor of Scholtz [115] (c.f. Chapter 2.2.1), the supervisor's desired commands can be subsumed as defining goals and modifying plans. Within the terminology used in this thesis, a goal can be translated to one or more tasks that compose the mission, while a plan describes which robot executes which task and when. Commands on the action level, that directly manipulate a robot's state, i. e., teleoperation, belong to the role of the operator and are not covered by supervisor commands.

Having obtained SO, the supervisor can express requests to changes of the goals and plans by commands of the following types:

1. *Define new goals:* The supervisor can express what the robots are supposed to do by adding new tasks to the mission. This allows to either refine a mission description by adding sub-goals, or to extend the mission if new information arises, that demands for further actions.
2. *Delete goals:* If the supervisor is convinced that some tasks are obsolete, and their execution does not advance the mission, these tasks can be deleted.
3. *Modify goals:* The parameters of a task can be adapted to adjust how this task is executed by the robots.
4. *Manually assign tasks to robots:* In case the supervisor wants a specific robot to execute a specific task, this task can be manually assigned to that robot.
5. *Release robots from single tasks or task types:* If a robot does not fulfill a task satisfactory, it can be released either from the single task, or from all tasks of a specific type.
6. *Interrupt all autonomous actions of a robot:* The supervisor can interrupt the robot's autonomy, which can be used to switch to another interaction role, e. g., operator or mechanic, if this is necessary to safeguard a robot from a difficult situation. SO is sufficient for a supervisor to recognize that such a switch is necessary, however, for the new interaction role a different awareness about the robot's state is usually mandatory.
7. *Define a preferred task type for a robot or a group of robots:* If a robot is very good at a specific task type, the supervisor can advise this robot to preferably execute tasks of this type. Likewise, if a robot is not good at a particular task type, the supervisor can advise this

robot to defer tasks of this type. These commands can also be applied to a group of robots or even to all robots, which leads to a shift of the whole team's focus towards the preferred tasks.

8. *Group a set of tasks for joint execution:* A human can detect a structure of the tasks, that is not easily detected autonomously by the robots. In this case the supervisor can group the tasks according to this structure, to suggest joint execution of tasks that apparently belong together.
9. *Exclude consecutive execution of some tasks:* Similar to grouping of tasks, a human can easily detect if connections between two or more tasks can be excluded, e. g., because the distance between them is large, or the path is very risky. The supervisor can exclude such sequences from the solution, for supporting a time-extended planner to quickly find a solution, and to exclude bad solutions for planners with a short look-ahead.

Commands 1 - 3 subsume “defining goals” as a part of the supervisors’ tasks, because these commands can be used to define *what* the robots are supposed to do. Likewise, the commands 4 - 9 correspond to “modifying plans”, as these commands can be used to specify the next high-level actions of the robots, and hence their current plans.

Many situations that benefit from or require supervisor interactions could also be overcome by tuning the autonomous behavior. However, covering all possible situations is very unlikely. Based on SO, the supervisor can decide how to define goals and modify plans to support the robot team. More detailed interactions, like modifications to the robot's state space or teleoperation in general, are not covered by the above commands. However, these are interactions that belong to the interaction role of an operator and can only hardly be executed based on SO.

In the following, examples for the described supervisor commands are given for the reference scenarios of Chapter 2.1. This list can only be exemplary, because there is an arbitrary number of possible interactions for each scenario.

5.2.2 Examples for the Reference Problems

In this section, examples are given for supervisor commands in the reference problems of Chapter 2.1.1.

Urban Search and Rescue - RoboCup Rescue

In the USAR context, one important factor is that the robots spread over the whole area, instead of all searching for victims at the same place. However, due to the unstructured environment and the varying degree of disruption of the area, it is equally important that each robot works, if possible, in that part of the area, where its capabilities are best suited for. It does, for example, not make sense if a wheeled robot explores a rubble pile, while at the same time a tracked robot explores a non-destroyed room with flat flooring.

Referring to the commands of the previous section, concrete examples for the USAR scenario are:

- *Define new goals:* In some disaster scenarios, prior knowledge about victim locations is available. For example, it may be known that a meeting was taking place in the conference room of a collapsed office building, and therefore it is assumed that people are trapped in this

area. Based on this knowledge, the supervisor can create well-directed search tasks around the assumed victim locations, that require the robot team to start their search in this area, to increase the probability of quickly finding the victims.

- *Delete goals:* Sometimes victim hypotheses are generated based on false detections of some algorithms. For example, a radiator can emit human-like temperature, and therefore a verification task is generated for this false hypothesis. A supervisor who realizes this mistake can delete this task, to prevent that a robot wastes time on inspecting a hypothesis that is obviously a false positive.
- *Modify goals:* To verify a victim hypothesis, the robots calculate a position close to the potential victim, from where it is assumed that the victim is best visible. Having obtained SO, the supervisor can judge whether this position is on the one hand indeed suitable for collecting more information about the victim hypothesis, while it is on the other hand accessible to the robot without too much effort. If the supervisor feels that some other position is more appropriate for the robot to inspect the potential victim, this goal can be modified.
- *Manually assign tasks to robots:* Based on SO the supervisor can judge if the robots spread properly over the search area. If necessary, the supervisor can manually send specific robots to other locations, to improve the spread and therefore increase the team performance.
- *Release robots from single tasks or task types:* Partial system failures, e. g., malfunctioning sensors or actuators, can restrain a robot from properly executing some tasks. As an example, consider a robot, whose victim detection algorithm relies mainly on heat detection, but the thermal camera fails to work. As these kinds of errors belong to the supervisor's knowledge about *robot health* as a part of SO, and the consequences of the defect are covered by the *task success potential*, the supervisor can forbid the robot to search for victims, because these tasks cannot be fulfilled by this robot satisfactorily. Instead, the robot can, e. g., map the environment, or transport water or health kits to victims that were detected, but not yet evacuated.
- *Interrupt all autonomous actions of a robot:* In case the supervisor notices that a robot has trouble with the current task, e. g., to climb up a stair, all autonomous actions of the robot can be interrupted to let an operator recover the robot from this situation. After the robot is recovered, the supervisor can switch the robot back to autonomous mode.
- *Define a preferred task type for a robot:* In a heterogeneous robot team, one robot may be much faster or more precise in performing specific tasks than the other robots in the team. If, for example, a robot has a sophisticated manipulator, this robot should favor to provide health kits to the victims, and leave the search tasks to other robots with inferior manipulation abilities.
- *Group a set of tasks for joint execution:* In case the supervisor recognizes the structure of several rooms, it makes sense to group all tasks in each room for joint execution respectively. In that way, each room is handled by a single robot, and not potentially by several robots together.
- *Exclude consecutive execution of some tasks:* The sequence of two tasks can be excluded, if there are other tasks on the way between these two tasks, or if the shortest way between these two tasks is risky, e. g., a robot has to climb a stair.

Urban Search and Rescue - DARPA Robotics Challenge

Because the structure of the mission in the DRC is entirely different to the other reference scenarios - the tasks are strictly sequential, and are not shared among several robots - not all supervisor commands can be applied reasonably. However, if the major tasks are modeled as sequence of dependent subtasks, most of the commands allow the supervisor to advise the robot and improve the overall performance.

- *Define new goals:* The definition of new goals can be useful to cope with unexpected situations, that occur either because of wrong robot behavior or for external reasons. For example, if the robot has to drive a car to a disaster site, and the car's engine stops suddenly, the supervisor can create a new task for the robot to restart the engine.
- *Delete goals:* The human can delete goals, if the robot has accomplished a subtask, but is not aware of this fact. Consider the task of removing debris blocking an entryway. If the robot has removed all blocking objects, the task can be deleted, even if some debris is remaining in the area, as long as the entryway is not blocked anymore.
- *Modify goals:* The modification of goals can be used to refine subtasks, for example to specify the area around a door that needs to be cleared.
- *Manually assign tasks to robots:* Because the tasks are sequential, usually only one task at a time is available to the robot. Manual task assignment however makes sense, in cases as described for *defining new goals*: after creating a task, the supervisor can assign this task to the robot.
- *Interrupt all autonomous actions of a robot:* The robot's autonomy can be interrupted to allow arbitrary teleoperation modes.

Robot Soccer

In the soccer scenario, conditions change very fast, and direct teleoperation is usually not an option. Instead, the supervisor can control the overall team behavior, the current tactic, and the parameters that specify how a specific role should be performed.

Examples in a robot soccer match for the previously defined supervisor commands are:

- *Define new goals / delete goals:* If the team is not successful in the match, the supervisor can change the team's tactic by deleting the roles of the current tactic and then adding the roles that compose the new tactic. Likewise, lots of different tactics can be defined by combining the available roles.
- *Modify goals:* Fine-tuning of the team's tactic can be done by adjusting each role's parameters, which describe the details of how a specific role shall be executed. For the striker the parameters describe, e. g., if the robot should prefer passing the ball to a teammate instead of kicking directly to the goal. The supporter's parameters describe where the robot should be positioned relative to the ball, and how far the robot is allowed to move towards the opponent and own goal. The goalkeeper's parameters determine the conditions when the robot should leave its goal area to clear a nearby ball.
- *Manually assign tasks to robots:* The goalkeeper role should always be performed by the same robot: on the one hand, because a robot that is able to pick up and throw the ball

is better suited for this role, and on the other hand because the rules require to have one dedicated goalkeeper. This can be achieved by manually assigning the goalkeeper role to this specific robot.

- *Release robots from single tasks or task types:* To prevent that any other robot than the designated goalkeeper is assigned to this role, the supervisor can forbid all other robots to take over the goalkeeper role.
- *Interrupt all autonomous actions of a robot:* In case a robot is taken out of the match (e. g., because of a penalty, or because it needs to be serviced by a mechanic), the supervisor can interrupt all of the robot's autonomous actions. Thereby, it is not necessary to switch off the robot for ensuring that it does not try to participate in the match anymore.
- *Define a preferred task type for a robot:* If one robot is considerably better in walking and kicking than its teammates (either because the team is heterogeneous, or because other robots are suffering from technical problems), the supervisor can advise this robot to preferably play as striker, to increase the team's performance.

Production Hall Logistics

The production hall logistics scenario is situated in a rather structured and well defined environment, with a homogeneous robot team. Malfunctioning robots can be removed and repaired, therefore problems with individual robots having difficulties in specific areas, like in the USAR scenario, usually do not occur. However, the tasks that compose the mission are, from the point of view of coordination, comparable to those of a USAR mission: here, the tasks are also finite (they have a well-defined start and end), and frequently new tasks can emerge during the mission, while in the soccer scenario the roles can last over the whole match. Therefore, in a production hall logistics setup, similar problems can occur regarding coordination as in a USAR mission. Additionally, there is also a struggle for scarce resources for the different products that have to be manufactured, which is even more tightened if some products have hard delivery deadlines.

Examples, how the supervisor can interact with the team to increase the performance are:

- *Define new goals / delete goals:* To communicate the currently required amount of products to the robots, the supervisor can add and delete tasks to modify the product orders.
- *Modify goals:* If the delivery deadline for a product changes, the supervisor can modify the respective task to ensure a faster production of this product if necessary.
- *Manually assign tasks to robots:* Based on SO the supervisor can recognize situations of bad coordination among the robots, e. g., if many robots cue at one machine, while other machines are idle. In that case, the supervisor can assign some robots to other tasks, that rely on the currently idle machine, to resolve this situation.
- *Release robots from single tasks or task types:* If not enough resources are available to produce all ordered goods, the supervisor can prohibit all robots to work on products that do not have delivery deadlines, to ensure that the more urgent products are completed first.
- *Interrupt all autonomous actions of a robot:* The supervisor can interrupt the autonomy of a defect robot, to allow that this robot is being serviced by a mechanic. Similarly, if an unauthorized person enters the workspace, the supervisor possibly needs to interrupt all autonomous actions of all robots for safety reasons.

- *Define a preferred task type for a robot:* If the timely production of a specific type of goods is important, the supervisor can advise some robots to prefer tasks that contribute to these products.

5.3 Proposed Concept for Realizing Supervisor Commands

Usually, the robots plan and coordinate their actions autonomously. Commands like those defined in the previous section give a human supervisor the power to influence the behavior of the team or of individual robots. This distribution – the supervisor does not have to do repeatable work because the robots coordinate autonomously, but is enabled to intervene if necessary – reflects the strengths of humans and robots (c.f. Chapter 1.1). In the context of this thesis, no assumptions are made on the duty of a supervisor to intervene in certain situation, the focus here is on *how* it can be achieved that the robots respect the supervisor’s commands.

5.3.1 Background: Task Allocation

The research field of task allocation (TA) deals with the question of how to allocate the tasks q_i , that compose the mission, to the robots in the team, in order to approximate the optimal solution for a given objective function. The objective can be, e.g., to minimize the total time spent to accomplish a mission, to minimize the energy consumed by all robots, or to maximize a scenario-specific profit function. Already in this simple form, if the number of tasks exceeds the number of robots, this problem is known to be NP-hard [44]. Hence, most problems are too large for calculating the optimal solution within a reasonable amount of time. Furthermore, in a dynamic real world environment, the tasks can change over time or new tasks can emerge during the mission. A formal description and classification of the multi-robot task allocation (MRTA) problem can be found in [44].

In order to reduce the time to calculate a solution, and to improve the responsiveness of the team to changing situations, heuristics are used to approximate the optimal solution. In the most general case, robots and tasks are heterogeneous, which means that not every robot can work on each task, and some robots are better in executing specific tasks than others. Not necessarily all tasks are known prior to the mission, and new tasks may emerge during execution, either because of new findings or changes in the environment, or because the supervisor introduces new tasks. Usually, no model is available for predicting task changes or the time when new tasks emerge.

Usually, work on task allocation assumes to deal with fully autonomous teams of (homogeneous or heterogeneous) robots, with either loose or tight cooperation. The primary approaches to TA can be divided into three classes: centralized, market-based and behavioral approaches.

Centralized Approaches

If the mission can be described as an optimal assignment problem (OAP) [41], the optimal solution can be found, e.g., using the Hungarian method [72], in $O(mn^2)$ time (with m robots and n tasks). However, in dynamic environments, not all costs can be computed in advance, and therefore most problems cannot be described as OAPs. Hence this method is usually not applicable in practice.

Search tasks can be described as a multiple traveling salesman problem (TSP, [73]), which can be transformed into a mixed-integer linear program (MILP) for faster solving [109], but still the

problem remains that changes in a dynamic environment or mission updates require to recompute the solution.

The constraint optimization coordination architecture COCoA presented in [69] combines heuristic methods with a MILP formulation into an anytime algorithm, to solve complex problems with interdependencies between different goals.

For tasks that require tight cooperation between several robots, Parker developed an algorithm called ASyMTRe [98], that connects schema to enable the robots to accomplish a task together, that none of the robots could have fulfilled alone. The algorithm finds the optimal solution, given enough time. The distributed version of this algorithm trades off solution quality for robustness.

Market-based Approaches

Market-based approaches are usually variants of the contract net protocol (CNP) [119]. Here, the robots trade tasks for revenue, to maximize the team's overall utility. In the simplest form, this results in a greedy scheduler, like MURDOCH [43]. This solution is 3-competitive¹ to the optimal solution, which is the best possible performance bound if neither planning in advance nor task re-allocation is allowed [66]. With the M+ architecture [19], also task re-allocation is allowed, and the robots plan one task in advance to achieve a higher solution quality.

One of the first market-based approaches is TraderBots, described in [21]. In [23], this architecture is extended to robot leaders, that centrally optimize the allocation within subgroups of robots. TraderBots has further been extended to handle pick-up teams, which are dynamically formed heterogeneous robot teams, where each team member has only minimal knowledge about the teammates [62]. In [112], each robot maintains a rough schedule of its future actions, and inserts traded tasks into this schedule. Similarly, in [124], even sets of tasks can be traded among robots to be integrated into local plans, which further improves the solution quality. In Hoplites [65], also coordination between robots is considered, which can be either passive, with agents implicitly influencing others, or with agents trading tasks for active coordination. [79] use local auctions, that guarantee a non-overlapping assignment also without complete connectivity, but is limited by the fact that the number of tasks may not exceed the number of robots.

Behavioral Approaches

In the context of behavioral approaches, each robot selects its actions based on local information. Cooperation and coordination emerges usually implicitly.

First behavioral approaches were inspired by collective behavior of insects like ants and bees, without using explicit communication among the individual robots [71].

In ALLIANCE [95], the robots broadcast their current activities to their teammates. The agents are motivated to execute tasks based on impatience and acquiescence, which allows them to take over tasks from other robots. Therefore, this approach is robust against partial or total robot failures.

With STEAM [120], robots use shared plans and joint intentions, which enables also intentional teamwork among the robots.

Discussion of Task Allocation Approaches

Besides the primary approaches to the task allocation problem, there are further approaches that do not directly fall in one of the three categories. For example, distributed constraint optimization

¹ the utility of the solution is at least 1/3 of the optimal utility or better

Criterion	Centralized	Market-based	Behavioral
Value of objective function	++	+	-
Time to calculate solution	-	+	++
Required communication bandwidth	-	+	++
Dealing with partial or total robot failures	-	+	++
Handling of mission changes during runtime	-	+	++
Dealing with heterogeneous robots	++	++	-

Table 5.1: Comparison between the three basic approaches to task allocation.

(DCOP) using tokens can significantly reduce the communication overhead between the agents [113, 33]. Stochastic clustering auctions can be used for swapping whole sets of tasks between robots, which is shown to work well for large numbers of heterogeneous robots and tasks [129]. In general, task allocation can be approached from many different directions, the choice of the applied algorithm is dependent on several factors, like environmental conditions, size of the robot team, available communication bandwidth, and mission specific requirements.

A detailed comparison between the three primary approaches can be found in [22] for the example of the multiple traveling salesmen problem. It turns out that the solution cost is lowest with the centralized approach, and highest with the behavioral approach. However, if the team size increases, the required overall time (including computation time) with the centralized approach exceeds the time spent with the market-based approach. Therefore, if it is important to have the optimal solution, a centralized approach is best suited, if only low computation power and communication bandwidth are available, the behavioral approach is the best choice, and otherwise the market-based approach is a very good trade-off.

Centralized approaches require typically a high bandwidth, because every robot has to send its calculated costs to the centralized planner, which in turn sends back the allocated tasks. Market-based approaches require much less messages, however, each task announcement results in bids from several robots and announcements of the auction winners. Behavioral approaches often require no communication at all, each robot senses the actions of the teammates or the robots broadcast their current tasks.

Usually, most behavioral approaches can deal with partial or total robot failures. Also many market-based solutions allow to recover from failures. Centralized solutions, that calculate the solution at the beginning, require lots of efforts to handle these cases.

Also mission changes during runtime cannot be handled properly using centralized approaches. In most cases, the only way is to recompute the whole solution. For the Hungarian method to solve OAPs, a dynamic version is available, that is able to adapt a previously calculated solution to changed costs [81]. Behavioral and market-based approaches, that do not plan ahead in time, can easily manage mission changes. The market-based approaches, that additionally maintain local plans, have to explicitly consider these changes.

For all three approaches, there are methods that can be applied to heterogeneous robots. However, many behavioral approaches are specifically designed for robot swarms, and not for homogeneous robots, because this would contradict one of the main advantages, that these approaches are simple and lightweight.

The strengths and weaknesses of the three approaches are summarized in Table 5.1. It can be seen, that each of the three approaches can outperform the other two, depending on the weighting of the criteria.

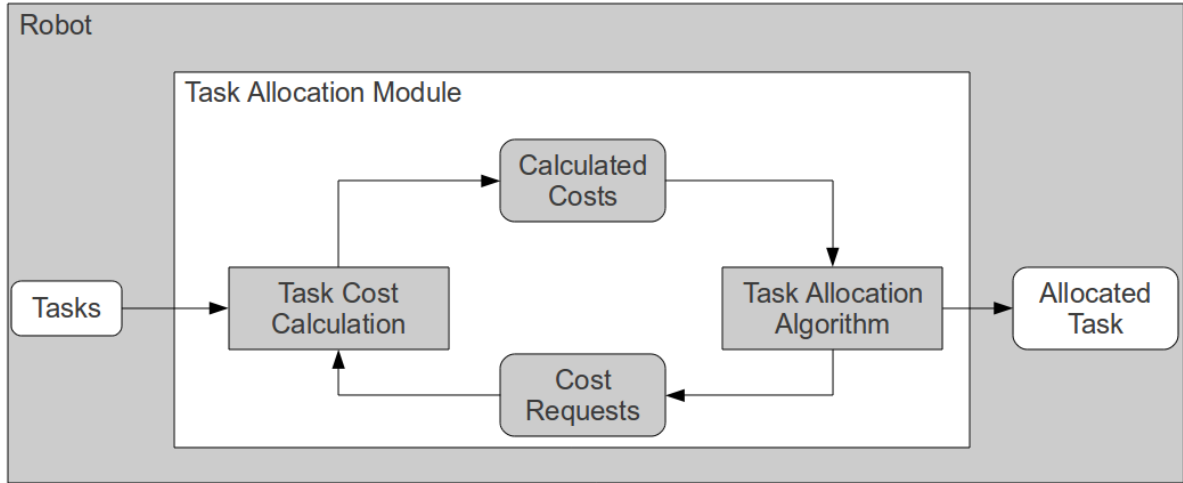


Figure 5.2: Functional parts of a task allocation module.

Assumptions on Task Allocation within this Thesis

On the one hand, the supervisor should have the power to influence the robots' behavior in many aspects. But on the other hand, this also implies, that the robots are expected to operate autonomously to a great extent. For example, although the supervisor could assign all tasks to the robots manually, this is not desirable, because it contradicts the specific capabilities of humans and robots as discussed in Chapter 1.1: manual task allocation would annoy the supervisor in the long run, and can usually be achieved better and faster by the robots autonomously. Therefore, one of the previously presented task allocation methods is used, and the supervisor is enabled to interact with this algorithm.

The possible commands from the supervisor to the robots are subdivided into two groups: commands that modify the mission details (add, delete, and modify tasks), and commands that affect the allocation of tasks to robots. As discussed in the previous paragraph, it makes sense to apply different TA methods dependent on the mission, the available robots and communication infrastructure. Therefore, the realization of the commands should be widely independent of the applied TA method, especially, it is not desirable to modify a specific TA algorithm. Moreover, the supervisor should not be required to know details about the currently used TA method.

To lay the foundations for a method to realize the supervisor commands independent of the applied TA method, the task allocation module is subdivided into two functional parts:

- *Task cost calculation*: This part calculates on request the cost to execute a specific task for a given setup of robot and environment, based on a predefined metric.
- *Task allocation algorithm*: This part determines, which task the robot shall execute, based on task costs. Internally, this part can optionally rely on behavioral schema, plan in advance, negotiate with other robots, or use a centralized planner.

This subdivision is visualized in Figure 5.2. The input data for the task cost calculation are the tasks that compose the current mission and the cost requests from the TA algorithm, while the output is a list of task costs for the corresponding requests. These task costs are the input for the other functional part, the TA algorithm, which generates as output data cost requests for the task cost calculation and an allocated task, which the robot needs to execute.

The task costs are used to reflect heterogeneities of the robots. For example, a fast wheeled robot has lower costs for traveling to a target position over flat ground than a slow tracked robot.

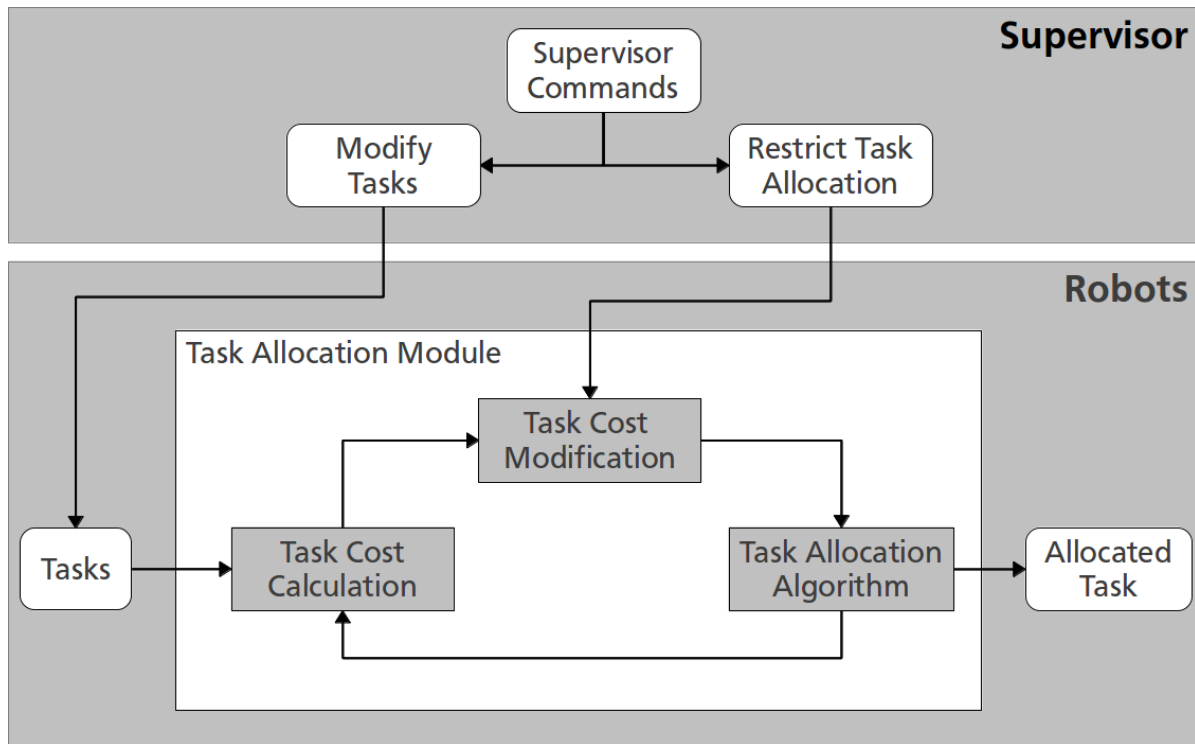


Figure 5.3: For instantaneous task assignment, only the tasks and the task costs for each robot are modified by the supervisor commands.

Accordingly, some robots are slower on certain terrains or cannot negotiate some terrains at all, therefore they have high or infinite cost for traveling to a target position that is only reachable via difficult terrain. Additionally, the costs can reflect the risk for executing a specific task, e. g., if a UAV has to land for achieving a task, this is more expensive than if a UGV only has to stop at the target position.

As it will be shown in the following sections, the supervisor commands are realized by modifying the input data for these two functional parts. This implies, that the algorithms within these functional parts do not need to be changed. For planning several steps in the future, some commands are translated into soft or hard constraints, which have to be respected by the TA algorithm. With instantaneous task assignment, modifying only tasks and task costs is sufficient, which means in particular, that an arbitrary cost-based TA algorithm can be used, e. g., a behavioral or market based approach.

5.3.2 Modification of Mission Details

The first part of the description of the supervisor role is *defining goals*. With the commands 1 - 3 as defined in Section 5.2, the supervisor can add, delete, and modify the tasks that compose the overall mission.

During a mission, the robots themselves can modify the mission details, e. g., to react to changes in the environment. Therefore, it can be assumed that the robot team can propagate and synchronize mission changes during runtime. Hence, mission modifications by the supervisor can in general be propagated among the robots with the same algorithm. The robots do not even need to consider that the changes originate from a human instead of a robot.

However, in some situations the supervisor needs more authority than the robots. For example, consider a robot that periodically recalculates a position, from where to inspect an object of interest. If the supervisor modifies this position, it will be eventually ignored by the robot, because the position is recalculated based on the same algorithm as before, and will most likely be close to the original position, that the supervisor wanted to overwrite. Therefore, it has to be stored which changes were introduced by the supervisor, and must not be modified by the robots autonomously.

Spoken in terms of the functional parts contributing to TA, the input which is taken for the cost calculation of the individual tasks is adapted by the supervisor. This is visualized in Figure 5.3 on the left side. The other parts of the TA are not affected by changes to the mission details.

5.3.3 Influence of Task Allocation – Instantaneous Task Assignment

The second part of the description of the supervisor role is *modifying plans*. With the commands 4 – 9 of Section 5.2 the supervisor is enabled to influence the task allocation. For instantaneous task assignment only 4 – 7 are relevant, because only one task at once is assigned, and sequences are not considered.

As stated in Chapter 5.3.1, it makes sense to choose different TA methods in different situations, dependent on the available communication infrastructure and computing power. The subdivision of the TA method into the functional parts as in Figure 5.2 allows to modify the input data for a TA algorithm with instantaneous task assignment, and thereby achieve an indirect modification to the allocation. Time-extended task assignment requires more efforts, which will be shown in the next section.

There are only a few assumptions the instantaneous TA method has to fulfill in order to be used with the presented method. The allocation has to be based on the cost for each robot to execute a task (or equivalently on utility). The calculated costs must be greater than zero, and the algorithm must not assign a task with infinite cost to a robot.

If these conditions are fulfilled, a new functional part can be inserted in between the task cost calculation and the TA algorithm. The input data are the calculated costs and the supervisor commands of type 4 - 7, the output data are modified costs for the tasks. These modified task costs are used as new input data for the TA algorithm. This extension to the original TA method is visualized in Figure 5.3.

In the following, Q is the set of tasks that describes the current mission. Further, $c(q)$ is the cost calculated by the task cost calculation for a robot to execute task $q \in Q$. The modified costs $c'(q)$ are determined based on the supervisor's commands as follows:

- *Manually assign tasks to robots:* If task q_1 shall be assigned to robot r , $c'(q_1)$ is set to 0, and $c'(q_i) = \infty \forall q_i \in Q \setminus \{q_1\}$ for robot r . For all other robots, $c'(q_1)$ is set to ∞ . This ensures that no other task except q_1 can be assigned to robot r because of the infinite costs. Furthermore, because for all other robots the cost to execute q_1 is ∞ , r is the best (and only) candidate to execute q_1 . Hence, task q_1 will be allocated to robot r .
- *Release robots from single tasks or task types:* To ensure that a specific task q or task of type P is not assigned to a robot r , the cost $c'(q)$ is set to ∞ for all these tasks. According to the assumptions on the TA algorithm, these tasks with infinite costs are not assigned to robot r .

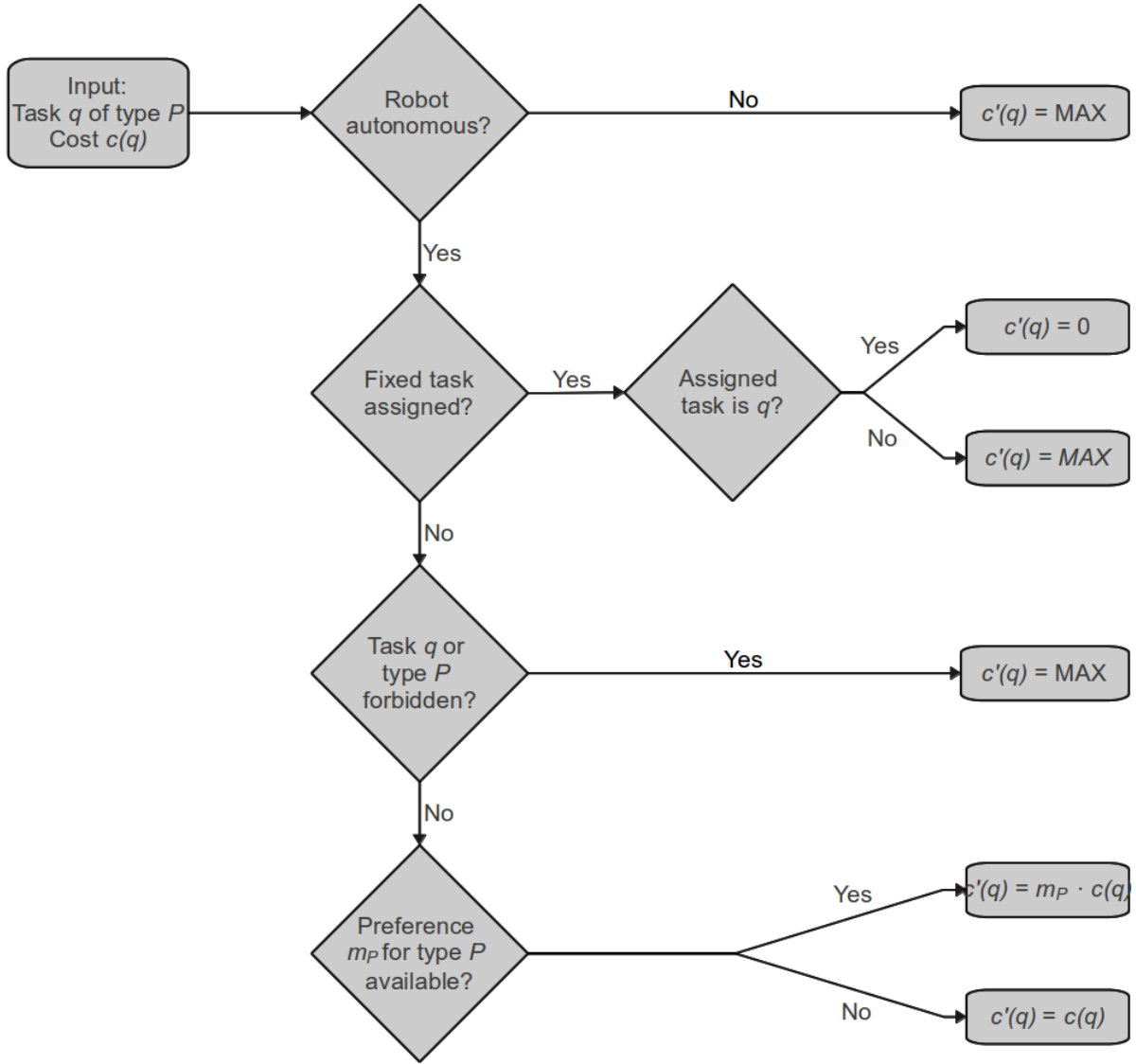


Figure 5.4: The algorithm to determine the modified task costs based on the original costs according to the supervisor commands.

- *Interrupt all autonomous actions of a robot:* All autonomous actions of a robot r can be interrupted by setting $c'(q) = \infty \forall q \in Q$. In that way, the TA algorithm must not assign any task to robot r .
- *Define a preferred task type for a robot:* To prefer or defer tasks of type P , the costs for all tasks $q \in P$ are scaled with a factor m_P . The size of m_P determines how much this task type is preferred or deferred, compared to other task types. On the one hand, if $m_P < 1$, the costs for each task of type P is reduced, and therefore these tasks are more likely to be assigned to robot r . On the other hand, if $m_P > 1$, the costs for the tasks of type P are increased, and are therefore less likely to be assigned to robot r .

The supervisor commands are applied hierarchically, which means that some commands are disregarded if a more important command is available. The hierarchy is sorted as follows:

1. *Autonomy:* The most important decision is, whether a robot is allowed to act autonomously at all. If the supervisor interrupts all autonomous actions, all other commands are disregarded.

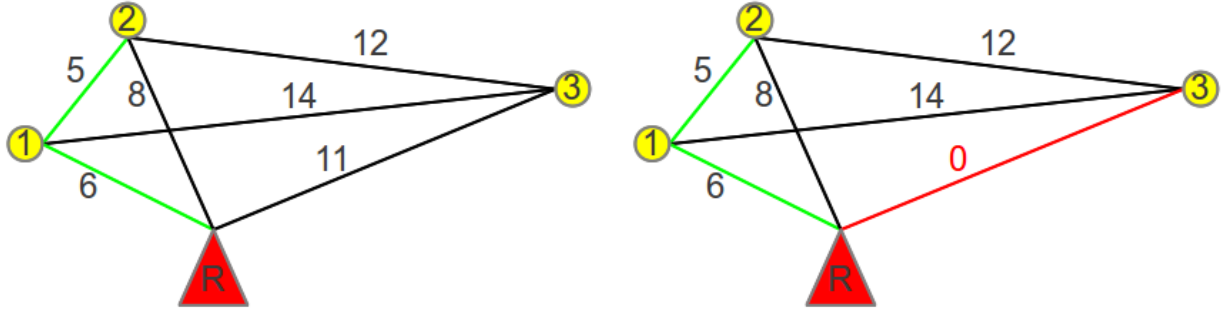


Figure 5.5: A simple example showing that cost modification is not sufficient for realizing the supervisor commands when dealing with time-extended task assignment. Even after reducing the cost to reach task 3 to 0, the optimal solution when planning 2 steps ahead is the sequence of task 1 and task 2.

2. *Fixed assignment*: A task that is manually assigned to a robot by the supervisor is executed, even if this contradicts an older command.
3. *Forbidden tasks*: Tasks or task types, that are declared as forbidden by the supervisor, cannot be allocated autonomously by a robot.
4. *Task type preferences*: As the last type of commands, task type preferences are applied to influence the task allocation.

For example, if the supervisor interrupts all autonomous actions of a robot, no task will be allocated to this robot, until it is set to autonomous mode again. Similarly, a robot that is assigned to a specific task will only work on other tasks, if either the supervisor releases the robot from the task, or if the task is accomplished. As soon as the robot is in full autonomous mode again, it complies with all previous supervisor commands, also with those that were given during the teleoperation phase or while the robot was working on a fixed assigned task. If the cost for executing a task q is not affected by any supervisor command, the originally calculated cost $c(q)$ is relayed to the TA algorithm. This hierarchy of the supervisor commands and the algorithm to determine the modified task costs are visualized in Figure 5.4.

New supervisor commands can overwrite older ones, e.g., if a task type has been initially forbidden, and then a new command requests to prefer these tasks with a certain factor, the first command is implicitly canceled. Based on the assumption that the supervisor usually wants the more recent commands to be executed, and does not want to struggle with messages about conflicting commands, this is a valid procedure. Likewise, a task can always be explicitly assigned to a robot by the supervisor, even if autonomous allocation is forbidden. This gives the supervisor the power to temporarily let the robot act against the older command, but later on, after this specific task is accomplished, the robot again obeys the original command.

5.3.4 Influence of Task Allocation – Time-extended Task Assignment

The method described in the previous section is not in all extends applicable when dealing with time-extended task assignment. As an example, consider the situation in Figure 5.5. The robot R (marked as a red triangle) has to visit tasks 1, 2 and 3 (marked as yellow circles). The edges are labeled with costs for the robot for traveling between the positions. If the robot plans two tasks

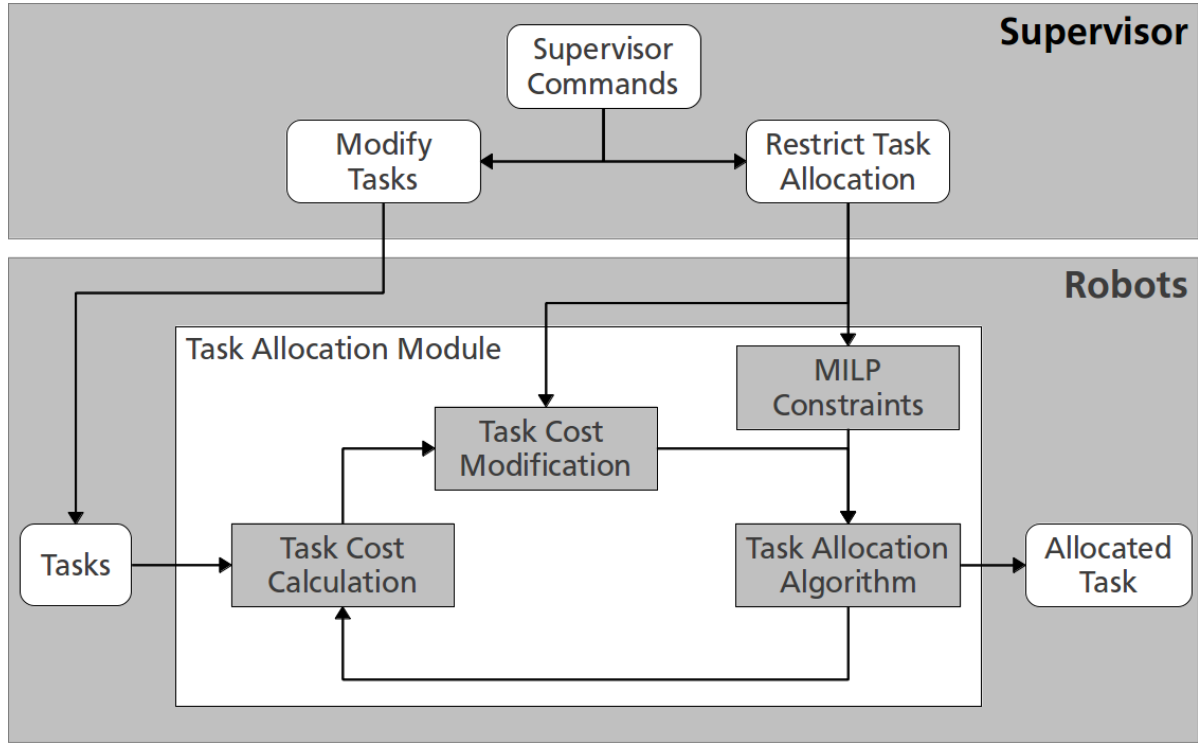


Figure 5.6: For time-extended task assignment, the tasks and the task costs for each robot are modified by the supervisor commands, and additional MILP constraints are added.

in advance, the optimal solution is to first visit task 1 and afterwards task 2, as marked in green on the left in Figure 5.5. If the supervisor wants the robot to first visit task 3, it is *not* sufficient to set the costs for this task to 0, because still the sequence of task 1 and task 2 is the optimal solution, as marked in green on the right in Figure 5.5.

This example shows, that for time-extended task assignment a different method is necessary for respecting all commands from the human. In the next paragraph a mixed-integer linear program (MILP) formulation for the URAR problem is presented. The supervisor commands are realized by adding constraints to the MILP and by modifying the task costs, as depicted in Figure 5.6. This MILP formulation and the integration are published by the author in [99]. The considered problem is to assign n robots to m tasks, some of them with timing constraints, over a sequence with finite planning horizon. A heterogeneous team R of robots is sent to explore a collapsed building, to find victims, and to supply the victims with water and first aid kits. A human supervisor H can support the team coordination of the robots from a remote location. An a-priori map of the building is available, where H can define a set of locations L . Each location $j \in L$ shall be examined by at least one robot $i \in R$. Whenever a robot detects a victim, it is added to the set V of victims. Each victim $k \in V$ needs to be revisited after a specific time for follow-up supplies, until it is actually rescued.

To explore a location $j \in L$, a robot $i \in R$ needs to reach the Δ^L surrounding of j (usually an ellipsoid), and scan it with a sensor, that is suitable to detect human evidence (e.g., a camera). Especially $\Delta^L.z$ (the height range of the visiting area) can typically be large, which implies that j can be either explored by a UAV (without the need to land there), or by a UGV. For supplying a victim $k \in V$ with water or health kits, a robot $i \in R$ has to approach the Δ^V radius of k , with

$\Delta^V < \Delta^L$. In particular, $\Delta^V.z$ is usually small, which forces a UAV to land at k . Robot i needs to stay at k for a certain time t_1^V for providing supplies to the victim.

In general, there are n robots, $i \in R$, $0 \leq i < n$, and a set of m tasks $j \in Q = L \cup V$, $1 \leq j \leq m$. The final number m of all tasks is not known in advance, because the supervisor H can define new tasks during the mission, and victims can be detected while the robots are working on the mission. Each robot $i \in R$ can only work on a single task $j \in Q$ at a time, but can sequentially execute one task after another. The cost κ_{ij} for robot i to execute task j is defined as the expected time required to accomplish the task. In particular this includes the time to reach a destination and the time a robot has to wait until it can do something else. Additionally, a revenue ρ_j is paid for completing task j .

For each robot $i \in R$, a cost matrix $K^i \in \mathbb{R}^{(m+1) \times m}$ is given, that defines the cost for executing task $j_2 \in Q$ after finishing task $j_1 \in Q$, with entries $\kappa_{ij_1j_2}$, $j_1 \in Q \cup 0, j_2 \in Q$. Entries κ_{i0j} describe the cost for executing task j starting with the current configuration (note that j_1 was defined to be between 0 and m).

Even though the MILP formulation is demonstrated for the USAR scenario, it can also be applied to other examples, as long as costs can be calculated for executing a task from a given start configuration. The timing constraints allow to describe that a task may not be started before a certain time t_{\min} and has to be finished earlier than a time t_{\max} . This allows to use the same MILP formulation also for the production hall logistics example. The costs model the time required to pick up, transport, and place the material, and to wait until it is processed if necessary. Delivery deadlines can be applied with a time limit t_{\max} . If some raw material is not available, but is expected to arrive at a given time, this can be modeled using t_{\min} , because tasks involving this material cannot be started before the material is available.

MILP Formulation

In order to formulate the problem as a mixed-integer linear program, binary variables x_{ij} are defined to equal 1 if task $j \in Q$ is assigned to robot $i \in R$, and zero otherwise. The robots can plan a fixed number of p tasks in advance. Each task can be assigned to exactly one robot, or being not assigned at all (which can happen if $m > p \cdot n$). This leads to the following constraints:

$$\sum_{i \in R} x_{ij} \leq 1 \quad \forall j \in Q \quad (5.1)$$

$$\sum_{j \in Q} x_{ij} \leq p \quad \forall i \in R \quad (5.2)$$

To account for the order of tasks, that are assigned to the robots, binary variables y_{ijk} are introduced, that indicate if task j is the k -th task for robot i :

$$\sum_{k=1}^p y_{ijk} \geq x_{ij} \quad \forall i \in R, \forall j \in Q \quad (5.3)$$

The robots start with an empty allocation, but may not explicitly be idle later in the schedule, which results in:

$$y_{i00} = 1 \quad \forall i \in R \quad (5.4)$$

$$y_{i0k} = 0 \quad \forall i \in R, \quad \forall 0 < k \leq p \quad (5.5)$$

No robot can have more than one task in the same slot of its schedule.

$$\sum_{j \in Q} y_{ijk} \leq 1 \quad \forall i \in R, \quad 0 \leq k \leq p \quad (5.6)$$

Furthermore, a robot can only have a task k in its schedule, if it also has a task $k - 1$.

$$\sum_{j \in Q \cup \{0\}} y_{ij(k-1)} \geq \sum_{j \in Q} y_{ijk} \quad \forall i \in R, \quad 1 \leq k \leq p \quad (5.7)$$

Given these constraints, the objective function is defined as the sum of all costs, that arise for executing the assigned tasks in the given order, and subtract the revenue earned for accomplishing the assigned tasks:

$$\sum_{i \in R} \sum_{j_1 \in Q \cup \{0\}} \sum_{j_2 \in Q} \sum_{k=1}^p y_{ij_1 k-1} \cdot y_{ij_2 k} \cdot \kappa_{ij_1 j_2} - \sum_{i \in R} \sum_{j \in Q} x_{ij} \rho_j \quad (5.8)$$

In this form, the objective function is non-linear, because of the product $y_{ij_1 k-1} \cdot y_{ij_2 k}$. This is resolved by introducing binary variables $z_{ij_1 j_2}$, that indicate that task j_2 follows task j_1 in the schedule of robot i .

$$z_{ij_1 j_2} \geq y_{ij_1 k-1} + y_{ij_2 k} - 1 \quad (5.9)$$

$$\forall i \in R, \quad \forall j_1 \in Q \cup \{0\}, j_2 \in Q, \quad \forall 0 \leq k \leq p$$

The resulting MILP with linear objective function is given by:

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in R} \sum_{j_1 \in Q \cup \{0\}} \sum_{j_2 \in Q} z_{ij_1 j_2} \cdot \kappa_{ij_1 j_2} \\ & - \sum_{i \in R} \sum_{j \in Q} x_{ij} \rho_j \\ \text{subject to} \quad & \text{Constraints (5.1) – (5.7) and (5.9)} \end{aligned} \quad (5.10)$$

So far, the MILP only describes the multi-agent scheduling problem without any timing constraints. However, the victim tasks have timing constraints. Each task has parameters $t_{j_{\min}} \geq 0$ and $t_{j_{\max}} \leq \infty$, that describe the earliest and latest time the task can be accomplished without penalties. Based on this, the variable \bar{t}_j models the time when task j is scheduled to be completed. The matrix E with entries $\eta_{ij_1 j_2}$ describes the time for each robot to execute task j_2 after executing task j_1 . This matrix may be equal to the cost matrix K , but this is not required, as K will be used later for modeling inputs from a human supervisor.

Additional variables p_{1j} and p_{2j} reflect if a task is scheduled too early (and hence the robot has idle time), or too late, given the time constraints of all tasks.

$$p_{1j} \geq t_{j_{\min}} - \bar{t}_j \quad \forall j \in Q \quad (5.11)$$

$$p_{1j} \geq 0 \quad \forall j \in Q \quad (5.12)$$

$$p_{2j} \geq \bar{t}_j - t_{j_{\max}} \quad \forall j \in Q \quad (5.13)$$

$$p_{2j} \geq 0 \quad \forall j \in Q \quad (5.14)$$

\bar{t}_j are modeled as recursive constraints based on the current schedule.

$$\bar{t}_j \geq z_{i0j} \cdot \eta_{i0j} + p_{1j} + t_{\text{now}} \quad \forall i \in R, \quad \forall j \in Q \quad (5.15)$$

$$\bar{t}_{j_2} \geq (\bar{t}_{j_1} + p_{1j_2} + \eta_{ij_1j_2}) \cdot z_{ij_1j_2} \quad \forall i \in R, \quad \forall j_1, j_2 \in Q \quad (5.16)$$

Equation 5.16 is non-linear, but because the variables $z_{ij_1j_2}$ are binary, it can be replaced by:

$$\begin{aligned} \bar{t}_{j_2} &\geq \bar{t}_{j_1} + p_{1j_2} + \eta_{ij_1j_2} - M \cdot (1 - z_{ij_1j_2}) \\ \bar{t}_{j_2} &\geq 0 \quad \forall i \in R, \quad \forall j_1, j_2 \in Q \\ &\text{with } M = t_{\text{now}} + \rho_{j_2} \cdot n \cdot p \end{aligned} \quad (5.17)$$

So far, the time is only modeled for tasks that are also scheduled. The following constraints model the best possible time for a task, that is not yet scheduled.

$$\bar{t}_{j_2} \geq (1 - \sum_{i \in R} x_{ij_2}) \cdot \min_{i \in R} \max_{j_1 \in Q} ((\bar{t}_{j_1} + \eta_{ij_1j_2}) \cdot x_{ij_1}) \quad (5.18)$$

Also this constraint is non-linear, on the one hand because of the min and max, and on the other hand because of the product between the variables. To linearize this constraint, more variables \hat{t}_{ij} are introduced, that model the time when task j could be finished, if it would be executed after the end of robot i 's schedule:

$$\begin{aligned} \hat{t}_{ij_2} &\geq \bar{t}_{j_1} + \eta_{ij_1j_2} - M \cdot (1 - x_{ij_1}) \\ \hat{t}_{ij_2} &\geq 0 \\ &\text{with } M = t_{\text{now}} + \rho_{j_2} \cdot n \cdot p \end{aligned} \quad (5.19)$$

With this, Equation 5.18 is now reduced to:

$$\bar{t}_j \geq \min_{i \in R} \hat{t}_{ij} - M \cdot \sum_{i \in R} x_{ij} \quad \forall j \in Q \quad (5.20)$$

However, Equation 5.20 is still non-linear because of the min. To get rid of this non-linearity, binary auxiliary variables d_{ij} are introduced, that represent if \hat{t}_{ij} is the minimum value for a specific task $j \in Q$, and variables \tilde{t}_j , that represent the minimal value. With this, Equation 5.20 resolves to the following constraints:

$$\tilde{t}_j \leq \hat{t}_{ij} \quad \forall i \in R, \quad \forall j \in Q \quad (5.21)$$

$$\begin{aligned}\tilde{t}_j &\geq \hat{t}_{ij} - M(1 - d_{ij}) \quad \forall i \in R, \quad \forall j \in Q, \\ M &= t_{\text{now}} + \rho_j \cdot n \cdot p\end{aligned}\tag{5.22}$$

$$\sum_{i \in R} d_{ij} = 1 \quad \forall j \in Q\tag{5.23}$$

$$\bar{t}_j \geq \tilde{t}_j - M \cdot \sum_{i \in R} x_{ij} \quad \forall j \in Q\tag{5.24}$$

Exceeding the time constraints is penalized in the objective function. Idle time for waiting until a task can be started (p_{1j}) is penalized with a factor α_1 . The time a task is accomplished too late (p_{2j}) is penalized with a factor α_2 . Typically, $\alpha_2 \gg \alpha_1$, because not meeting a task's constraints is more critical than idle time. The final objective function is defined as:

$$\begin{aligned}\sum_{i \in R} \sum_{j_1 \in Q \cup \{0\}} \sum_{j_2 \in Q} z_{ij_1 j_2} \cdot \kappa_{ij_1 j_2} - \sum_{i \in R} \sum_{j \in Q} x_{ij} \rho_j \\ + \sum_{j \in Q} (p_{1j} \cdot \alpha_1 + p_{2j} \cdot \alpha_2)\end{aligned}\tag{5.25}$$

Overall, the constraints matrix quickly gets very large. Both, the number of variables (the columns of the matrix) and the number of constraints (the rows of the matrix) grow linearly with the number of robots and with the planning horizon and quadratically with the number of tasks. Because the constraints depend on the variables, the overall size of the matrix (and therefore the required calculation time) grow much faster. For a given problem, the number of robots and tasks cannot be adjusted, therefore, reducing the planning horizon is the only possibility to reduce the problem size for solving the whole problem in reasonable time.

Human Supervision

The user interface allows the supervisor to express intuitive constraints, which are then automatically translated into constraints for the MILP. In the following possible translations for the commands 4 – 9 into MILP constraints are presented.

Manually assign tasks to robots:

Strict assignments are achieved by adding hard constraints to the MILP. Given there is a single task j that the human wants to be accomplished, the following constraint will be added:

$$\sum_{i \in R} x_{ij} = 1\tag{5.26}$$

So far, this does not say anything about *when* the task has to be executed. If the task has to be among the first N tasks in the schedule of a robot with $N < p$, the following constraint is added:

$$\sum_{i \in R} \sum_{k=1}^N y_{ijk} = 1\tag{5.27}$$

If the supervisor wants instead that the task is executed within a specific time limit, the deadline $t_{j\max}$ is adapted accordingly, and a hard constraint requiring the task to be finished before the deadline is added:

$$\bar{t}_j \leq t_{j\max} \quad (5.28)$$

However, in the worst case, adding hard constraints can render the problem infeasible, e.g., if deadlines are chosen too strictly. Thus, inputs by the supervisor are more safe to be integrated as soft constraints. To achieve that task j is preferably part of the optimal solution, either the revenue ρ_j can be raised or the cost κ_{ikj} to execute this task can be lowered $\forall i \in R, k \in Q \cup 0$. A soft constraint on a time limit can be added by defining a deadline $t_{j\max}$, but without putting a hard constraint on \bar{t}_j .

Release robots from single tasks or task types:

In case a group $R_1 \subset R$ of robots shall not be allowed to execute a group of tasks $Q_1 \subset Q$, the following constraints are added to the system:

$$x_{ij} = 0 \quad \forall i \in R_1, \forall j \in Q_1 \quad (5.29)$$

Interrupt all autonomous actions of a robot:

To forbid that any task $j \in Q$ is assigned to robot i , the following constraints are added:

$$x_{ij} = 0 \quad \forall j \in Q \quad (5.30)$$

Define a preferred task type for a robot or a group of robots:

Global priorities can be modeled by modifying the revenue values ρ_j . Let $Q_1 \subset Q$ be the tasks that the robots should focus on, and $Q_2 = Q \setminus Q_1$. To shift the robots' focus towards tasks in Q_1 , the revenue for these tasks is scaled by a factor $a > 1$. If the assignment of tasks in Q_2 shall be highly unlikely, the revenue can be set to 0, i.e., $\rho_j = 0 \quad \forall j \in Q_2$. Equivalently, the costs $\kappa_{ij_1j_2}$ for executing tasks $j_2 \in Q_1$ can be reduced by a factor $0 < b < 1$, or raised by a factor $c > 1$ for tasks in Q_2 respectively.

In case the tasks in Q_2 shall be completely excluded, a hard constraint can be added:

$$x_{ij} = 0 \quad \forall i \in R, \forall j \in Q_2 \quad (5.31)$$

If this focus shall only be defined for a subset of robots (e.g., robots of a specific type, with a specific capability, or only for a single robot), these factors and constraints are added only for the affected robots. Adapting the revenue values is not possible in that case, because they are common for all robots.

Group a set of tasks for joint execution:

As an example, consider the scenario in Figure 5.7. If two robots are available to work on these tasks, it is apparently a good solution to execute the tasks in the left room as one group, and tasks in the right room as a second group. To model this command as a soft constraint, the costs to execute sequences within a group J are lowered. Depending on how strong this soft constraint is intended to be, $\kappa_{ij_1j_2}$ is scaled with a factor $0 < a < 1, \forall i \in R, j_1, j_2 \in J$. The smaller the factor a is chosen, the more likely will a robot that works on one of these tasks also execute the other tasks in the group J . To model the same request as a hard constraint, the following constraints with new binary variables g_i are added to the system:

$$k \cdot g_i \leq \sum_{j_1, j_2 \in J} z_{ij_1j_2} \quad \forall i \in R \quad (5.32)$$

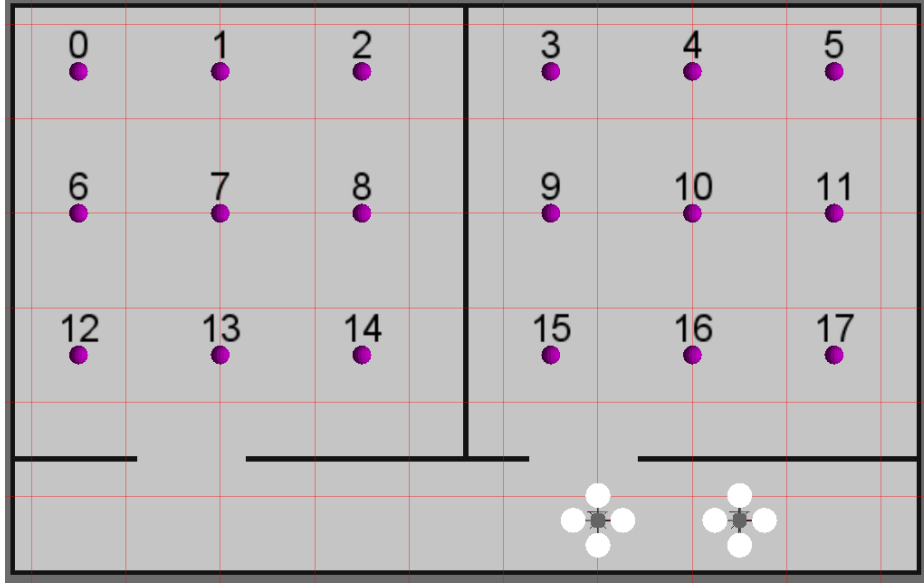


Figure 5.7: Example scenario for joint exploration of a small indoor environment.

$$\sum_{i \in R} g_i = 1 \quad (5.33)$$

This constraint requires, that one robot has at least k tasks of J in its schedule. Parameter k must not be larger than the planning horizon p , otherwise the problem is infeasible.

Exclude consecutive execution of some tasks:

In the example in Figure 5.7, connections between most tasks in the left and the right room are apparently not very effective. To exclude a specific sequence $j_1 j_2$ from the set of feasible solutions, the following constraint is added:

$$z_{ij_1 j_2} = 0 \quad \forall i \in R \quad (5.34)$$

As soft constraints (the sequences are still allowed, but unlikely to be selected), the costs $\kappa_{ij_1 j_2}$ for executing j_2 after j_1 can be scaled by a factor $b > 1$.

Hard constraints introduced by the supervisor can make the problem infeasible. The easiest way to resolve this is to simply reject such constraints. Another option is to compute an irreducible inconsistent subsystem (ISS) that causes the infeasibility, and either remove these constraints or present them to the supervisor for further inspection. However, this requires the supervisor to have a detailed knowledge about the model, which is not assumed to be the case. Instead of removing constraints, slack variables can be used to meet the constraints as good as possible. In all cases, some commands of the supervisor cannot be addressed properly. Therefore, soft constraints should be preferred over hard constraints if possible, even though hard constraints result in a larger speed up of the computation time.

5.4 Advantages of the Proposed Approach

The presented approach for interactions between a human supervisor and an autonomous robot team is limited to the described interactions on a high level. This means in particular, that close interactions like teleoperation are not supported. However, the supported interactions correspond

to the strengths of humans and robots as presented in Chapter 1.1, and are therefore well-suited especially for team interactions. The approach of interacting at a higher level allows the supervisor to have much less knowledge about the robots compared to the required knowledge for teleoperation, e. g., regarding the details of the locomotion or algorithms applied for task allocation, but still it can be assumed that a well-trained supervisor can support the robots much better than a novice. Besides these limitations, the presented approach implicates several crucial advantages.

Adequacy for SO-based Decisions

The developed supervision concept is adequate to realize supervisor commands that result from decisions a supervisor can take based on SO. This includes the definition and refinement of the team's goals as well as the modification of plans. For more detailed interactions, some other interaction mode and another knowledge base (i. e., SA instead of SO) is required.

Extension to Autonomy

Instead of replacing parts of the robots' existing software for autonomous operation, the presented method extends existing parts. The supervisor is enabled to introduce commands that refine goals or modify plans, but technically not required to intervene. In case the supervisor does not give any command, the autonomy of the robots is not affected at all, and hence the robots can still proceed towards their mission goal. According to [54], this is one requirement to achieve a good performance of a human-supported robot team.

Adaptable Level of Autonomy

Using the policy system of the previous chapter allows to adapt dynamically, which queries are sent to the supervisor, or which queries are answered autonomously. This allows to adapt the robots' LOA.

Invariance to Application Scenario

Because the developed methods operate at a very abstract level of the robot control software, they do not make use of specific characteristics of any mission type. This means, that they are not restricted to a specific application, but generally applicable to various different scenarios. Some examples were already given in Section 5.2, furthermore some experiments and results are presented in Chapter 7.4. Additionally, the application to other scenarios, e. g., monitoring and surveillance or waste disposal, is also possible.

Invariance to applied Task Allocation Method

The developed method modifies the input data for a TA algorithm, but does not change the actual algorithm. This implies, that for instantaneous task assignment an arbitrary TA method can be used, e. g., a centralized, market-based, or behavioral method. The solution for time-extended task assignment relies on a MILP formulation, however, different solvers can be used for calculating the solution. This further increases the applicability to different problem classes, because an appropriate TA method can be selected and combined with the supervision concept. With instantaneous task assignment, the TA method can even be selected or changed during runtime, which gives a well-trained supervisor even more flexibility to enhance the team's performance.

Use by Automated Supervision Agents

The supervision concept can not only be used by human supervisors, but can even be applied to automated supervision agents. For example, if a specific SO event is known to always cause the same supervisor command, this can be automated by a supervision agent, that sends the respective command to the robots. Furthermore, if the robots are monitoring their own health, they can use supervisor commands to restrict their actions, if their abilities are limited due to defects or other problems. An example of an autonomous supervision agent is presented in Chapter 7.4.4.

6 User Interface

According to Endsley [30], two aspects are important for enabling a human to assess a situation: which data is provided, and how the provided data is presented. The first issue was addressed in Chapter 3 and Chapter 4, while the second aspect will be shortly addressed in this chapter. Furthermore, in addition to data representation, a user interface must provide means for expressing commands as defined in Chapter 5.

6.1 Background: Interface Design

Endsley lists some interface features that support a human in obtaining situation awareness [30]. The interface should not only display raw data, but rather enhance it with interpretations and predictions for future events, to reduce the operator's workload and support unexperienced users with building mental models of the system under control. Critical elements should be presented more salient than regular information to help manage the operator's attention. The data should be displayed in relation to the current goals, and not in relation to the technological aspects of the system. Furthermore, the data should be reduced as much as possible, by fusing information and filtering information that is not required for the current needs of the operator.

According to Norman [89], the information a human can keep in mind, called “knowledge in the head” is very limited, hence it is important that an interface presents external information, called “knowledge in the world”, to remind the supervisor of the state of the system and the open tasks.

Goodrich and Olsen define seven design principles for efficient human robot interaction, that address not only data representation, but also human input [47]. Like Endsley, they also state that the interface should support the attention management of the human. In accordance with Norman, the interface should provide means to support the human's memory. Regarding user inputs, they state that the human should be enabled to manipulate the world or the relationship between the robot and the world, instead of directly manipulating the robots. Furthermore, the interface should switch automatically to the current autonomy level and control mode. This is consistent with the report of an operator of one of the ground robots deployed after the nuclear disaster in the Fukushima power plant. In this deployment, manual mode switching was often critical, because it had to be performed during unstable tasks [49].

Especially in the USAR domain, much research has been done on user interfaces, that allow one operator to remote control one robot, e. g., [64, 86]. These strongly rely on video- and map-data, that needs to be sent in real-time from the robot to the user interface. On the one hand, this allows to accurately control a robot even in unstructured and complicated environments, but on the other hand, those interfaces cannot be extended easily to control more than one robot simultaneously, and require high bandwidth, which is often not permanently available in real-world scenarios. Nielsen et al. demonstrate the superiority of an integrated 3D interface over interfaces consisting of several separate windows [86]. These types of interfaces are inspired by first person shooter games, and therefore people with experience in these types of games can be expected to learn faster how to control the robot.

When controlling more than one robot, the two frequently applied interaction modes are sequential operation (e. g., [63]) and playbook interfaces (e. g., [80]). Also these types of interfaces are often inspired by computer games, for example strategy games, where the user has to control

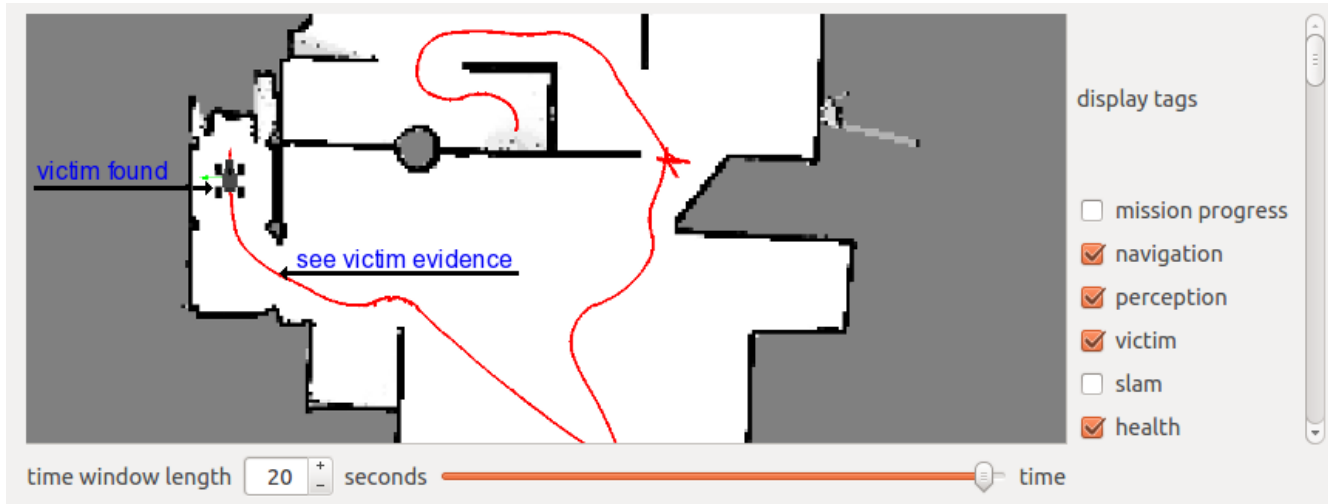


Figure 6.1: An example for a graphical representation of events in a USAR mission.

a large number of agents simultaneously. Based on four experiments, Goodrich et al. conclude that adaptive or adjustable autonomy can improve the performance in both cases, with sequential operation and with playbook management [46].

6.2 Interfaces for the Developed Concepts

The previous section shows, that developing a good user interface is a difficult task. A general interface, that is applicable to all targeted applications and adheres to the design considerations for efficient user interfaces is probably impossible to develop.

The focus of this thesis is *which* data is communicated between the supervisor and the robot team. Therefore, the applied interfaces are in most aspects only functional, and are difficult to use by non-experts.

For the event based communication, the criticality levels allow different representations at a user interface. For example, warning should be highlighted, e.g. with a different color, compared to information events. Errors can even be represented as pop-ups, because it is crucial for the supervisor to be fully aware of these events. Storing all arriving events for a specific time can be used, for example, to re-obtain SO after a period of inattention by the supervisor, if the events are displayed along a timeline. Furthermore, filtering and sorting according to the tags can also support the supervisor in quickly assessing the situation.

An exemplary graphical user interface for a USAR mission is shown in Figure 6.1. The central element is the map learned by the robot. Furthermore, the robot's path is displayed. Events are displayed at the location they occurred. By clicking on an event, the payload can be displayed, for example images of the victim. On the right, the user can select the events to display based on a list of all available tags. In the bottom the user can slide back and forth in time, and define the maximum age of displayed events as the maximum time window. In this specific example, if the supervisor would either increase the time window or go back in time, further events would appear, for example a *stuck* event at the narrow passage, where the robot's path shows some back and forth movements. In this example, only the central element with the map is specific to the concrete mission. Similar interface can also be designed for displaying events for other scenarios.

	robot_p2_rev-13710		robot_p3_rev-13710	
Autonomous	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	
Type: striker	prefer	defer	prefer	defer
Type: supporter	prefer	defer	prefer	defer
Type: goalie	prefer	defer	prefer	defer
Type: libero	prefer	defer	prefer	defer
Type: assiststriker	prefer	defer	prefer	defer
Task: striker_1	<input type="checkbox"/> assign	<input type="checkbox"/> forbid	<input type="checkbox"/> assign	<input type="checkbox"/> forbid
Task: goalie_2	<input type="checkbox"/> assign	<input type="checkbox"/> forbid	<input checked="" type="checkbox"/> assign	<input type="checkbox"/> forbid
Task: supporter_3	<input type="checkbox"/> assign	<input type="checkbox"/> forbid	<input type="checkbox"/> assign	<input type="checkbox"/> forbid
Task: supporter_4	<input type="checkbox"/> assign	<input type="checkbox"/> forbid	<input type="checkbox"/> assign	<input type="checkbox"/> forbid

Figure 6.2: Functional interface used for the soccer scenario (part 1).

For supervisor inputs in the soccer scenario, a functional interface (Figure 6.2 and 6.3) is used. Figure 6.2 shows a list of all available task types and the role instances, that compose the current tactic. For each connected robot, the user can ask the robot to prefer or defer roles of a specific type. In the depicted example, the robot on the left maximally defers the goalkeeper role, because the other robot is the designated goalkeeper, and hence no other robot should execute a goalkeeper role. Furthermore, blue highlighted roles for the robots show the current allocation. The supervisor can manually forbid assignments or assign the robots to the roles, like the right robot is assigned to the goalkeeper role, using the checkboxes. The dialog in Figure 6.3 allows the supervisor to define details of the current tactic, like the roles that compose the tactic, including the exact

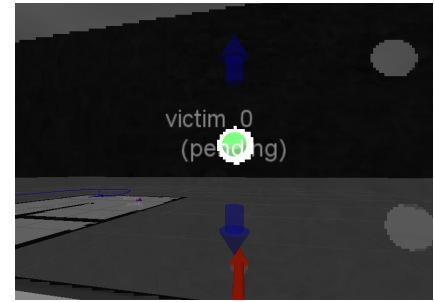
	priority	role	parameters
striker_1	 	striker	kick_set 0
goalie_2	 	goalie	home_x -2700 clear_x 0 clear_x_no_wifi 3000 clear_x_no_wifi_team -1200 kick_set 0 estimated_opp_kick_dist 3000 suspend 0
supporter_3	 	supporter	offset_x 500 offset_y 1200 min_x -2400 max_x 2400
supporter_4	 	supporter	offset_x -1000 offset_y 1200 min_x -2400 max_x 0

Save Tactic

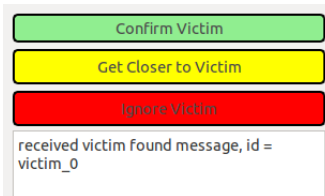
Figure 6.3: Functional interface used for the soccer scenario (part 2).



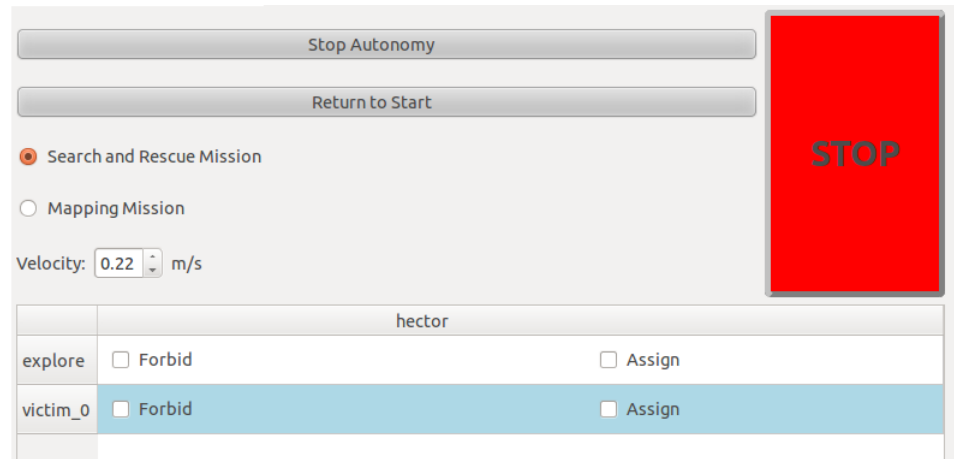
(a)



(b)



(c)



(d)

Figure 6.4: Functional interface used for the USAR scenario: (a) The main rviz window with the current map, robot position, and interactive markers for user input. (b) The image of the simulated thermal camera. (c) The rqt_gui plugin for the victim queries. (d) The rqt_gui plugin with an overview about open tasks.

parametrization. The parameters for the individual roles require the user to know how they have to be interpreted, and hence a novice user without system knowledge cannot efficiently use this interface. However, the interface is very general, and automatically detects the current roles and available parameters, and therefore can handle newly developed roles, without the need to adapt the interface. A more intuitive interface can, for example, be based on a visualization of the playing field, and let the supervisor interact like with a coach board.

Also for the USAR scenario an interface listing all tasks is available (Figure 6.4d). Like in the soccer interface, the currently allocated task is highlighted, and the supervisor can manually assign or forbid tasks. For the queries regarding the detected victims, a dialog as depicted in Figure 6.4c is used. Tasks can be added, deleted, and modified via a map representation (Figure 6.4a). All

tasks are visualized on this map (e. g., as arrows) and can be dragged by the supervisor. A menu allows further interactions, for example direct assignment of a task to a robot. This interface is more intuitive than the soccer interface, but is also not optimized for fast and efficient interactions.



7 Experiments and Results

7.1 Simulation of Autonomous Robots

Conducting experiments in a 3d-simulation has several advantages over using real robots. In a simulation the environment can be arbitrarily configured and controlled. This allows to set up larger and more versatile test scenarios than in a lab experiment. The conditions are repeatable, and therefore exactly the same experiment can be conducted several times. No hardware failures can bias the results, however, intended defects can be simulated on purpose. Therefore, tests over much longer time periods are possible. Moreover, in a simulation it is possible to use more different types and a larger number of robots, than are available in reality.

For these reasons, many experiments in this thesis are conducted in simulation. Further experiments and use cases are also demonstrated with real robots.

The simulation for the soccer robots is based on the multi-robot simulation framework MuRoSimF [40]. Two teams of three robots can be simulated in realtime, using a kinematic motion simulation. This implies that the robots cannot fall over or slide while walking. Instead of simulating a camera for each robot, the image processing step is skipped, and the robots' pose on the field, the position of the landmarks and of the ball are provided to each robot by the simulation. It is possible to add noise or other error models to this data, to imitate effects of an imperfect sensor processing.

The experiments for the USAR robots are conducted with the 3d simulation gazebo (<http://www.gazebosim.org>). Gazebo uses the *open dynamics engine* (ODE, <http://ode.org/>) for physics simulation and the *object-oriented graphics rendering engine* (OGRE, <http://www.ogre3d.org>) for rendering. It is able to simulate various different ground robots and aerial vehicles, equipped several different sensors like cameras and laser range finders. The simulator has a ROS integration, including a simulated clock to run simulations faster or slower than real-time. Test arenas like those described in Chapter 2.1.1 can be generated using an open source arena designer (http://ros.org/wiki/hector_nist_arenas_gazebo). This allows to quickly generate new environments or reproduce in simulation the layout of real test arenas.

7.2 Performance Metrics

Common tools for evaluating the quality of a human-machine interface are the situation awareness global assessment technique (SAGAT) [28, 29], the situational awareness rating technique (SART) [121], and the NASA task load index (NASA-TLX) [55]. The focus of all three is to evaluate the applied interfaces and the workload of the human operator. Therefore, neither of them is applicable here, because the presented methods focus on the communication between the robots and the supervisor, but not on the interface. Furthermore, the robots are assumed to be able to solve their mission completely autonomous, the supervisor can adapt the LOA and thereby regulate his own workload.

Some widespread metrics for evaluating human-robot interactions are introduced by Olsen and Goodrich [90]. They attempt to identify which parts of the system limit the overall performance. Task effectiveness (TE) describes the overall performance of the human-robot team and is mission



Figure 7.1: A scene from the semifinal match at RoboCup 2011 between Darmstadt Dribblers and Team Darwin.

dependent. It can, e. g., include time-based metrics, error metrics, and coverage metrics. Because the purpose of all methods presented in this thesis is to improve the overall team performance, TE is the main metric that is applied in the presented experiments. The metrics neglect tolerance, robot attention demand, free time, fan out, and interaction effort describe how much time a human has to spend with each robot to keep the performance above a given threshold, and how many robots a single human can operate. However, these metrics do not cover the simultaneous interaction with more than one robot. Furthermore, in this thesis it is assumed that the robots are able to solve their mission even without human support. Therefore, these metrics are not applicable in the context of this thesis.

This set of metrics has been further extended by Crandall and Cummings [17] by the metrics of attention allocation efficiency and switch times. These describe the time the supervisor needs to switch his attention from one robot to another. Also here the authors assume the sequential operation of all robots in the team. In contrast to these assumptions, the interaction types presented in this thesis involve several or all robots of a team. Therefore, also these metrics cannot be applied here.

7.3 Event Detection – Monitoring Trends in a Robot Soccer Match

To demonstrate simple and complex events, consider a robot soccer match. The efficiency of the current team strategy shall be monitored, to decide about potential tactic changes. The events are discussed for the semi final of RoboCup 2011 between the team Darmstadt Dribblers (referred to as *own team*) and the team Darwin (referred to as *opponent team*). An image of the match can be seen in Figure 7.1. Videos of the whole match can be seen on <http://youtu.be/RRelIBW0icc> (first half) and <http://youtu.be/FGnhRRPrf1Q> (second half and extension). The detected events for this match are visualized in Figure 7.2.

Simple events are generated when the own team scores a goal (E_1 , shown as blue circles in Figure 7.2) or when the opponent team scores a goal (E_2 , shown as red triangles). An *own run* is defined as the sequence (E_1, E_1, E_1) , that must not be interrupted by event E_2 . An *opponent run* is defined accordingly. Own runs are depicted as blue crosses, opponent runs as red stars. Runs are important to monitor, because they indicate a longer period of particularly good performance of one of the teams. In case of opponent runs, a change of the own tactic should be considered, to interrupt the success series of the opponent team. In the considered match, the own team had

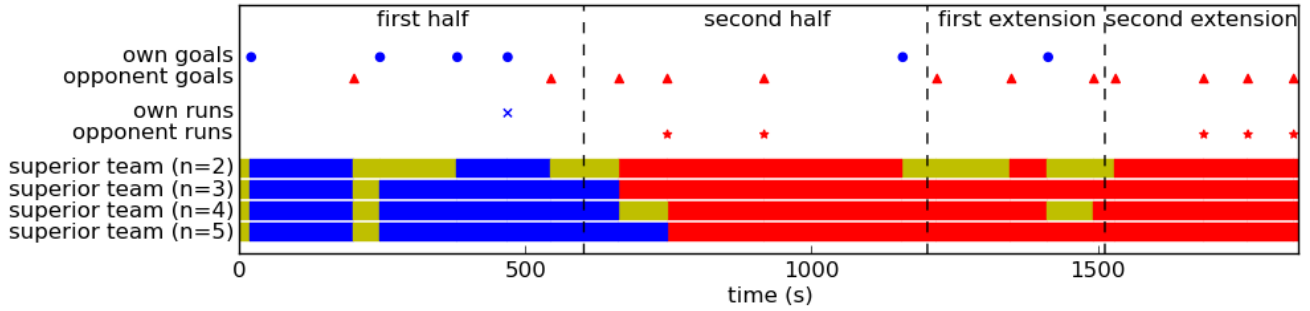


Figure 7.2: Goals scored in the semifinal match at RoboCup 2011 and resulting events. Own and opponent goals are simple events. Based on these simple events, the complex events of runs and team superiority can be detected based on the last n goals. Own superiority is marked in blue, opponent superiority in red, and balanced strength in yellow.

a run in the first half, and the opponent team had runs in the second half and in the second extension.

However, a team can win a match even without runs. The overall score of the match reflects which team is more successful since the beginning of the match. Using complex events, also the short term performance can be monitored based on the last n scored goals, which allows to adjust the duration of the monitoring period. For the last n goals, the number a of own goals, and the number b of opponent goals is counted, $n = a + b$. The own team is superior for the considered period if $a > b$. Likewise, if $a < b$, the opponent team is superior. Both teams are equally strong if $a = b$. In Figure 7.2, superiority of the own team is marked with a blue bar, superiority of the opponent team is marked with a red bar, and equal strength is marked with a yellow bar. The team performance was evaluated for $n = 2, 3, 4, 5$. A smaller n is more responsive, while a larger n is more stable against outliers. It can be seen, that the own team was superior during the first half, but the performance degraded during the second half.

For the performance analysis, also other data could be considered, e.g., the position of the ball on the playing field. The closer the ball is to the opponent goal, the better is the assumed performance of the own team. Hence, the ball position, sampled for example once a second and averaged over the last m seconds, can be used as another indicator for team superiority. The two performance indicator events, based on the number of scored goals and based on the ball position, can be fused, to a more fine grained event. If both indicators agree on one superior team, this results in a new event indicating strong superiority. Otherwise, if, for example, the ball is usually closer to the own goal, but the own team scored more goals, an event indicating weak superiority is generated. The goal is, to achieve strong superiority of the own team.

7.4 Human-Robot Interaction

7.4.1 Experiments in Urban Search and Rescue - Instantaneous Task Assignment

The experiments in this section address the basic version of human-initiated interactions described in Chapter 5.3 for instantaneous task assignment. The results of these experiments have been presented in [103].

These experiments were conducted with the simulator gazebo. The setup corresponds to Chapter 2.1.1. However, the arena is not partitioned into yellow, orange, and red parts, instead the whole arena is accessible to the autonomous robot. In the supervised mission, input from one human was allowed, but only using the commands described in Chapter 5.2. In particular, direct teleoperation was not allowed.

Because the aim is to enable a human supervisor to support autonomous robots, the results of a purely autonomous robot are compared to those of a supervised robot. The autonomous baseline is the solution by Team Hector Darmstadt [4], which won the “best-in-class autonomy award” and placed second in the overall rescue robot competition at RoboCup 2012.

Both, the autonomous and the supervised robot, used the same software. In particular, the same exploration strategy, the same victim detection, and the same task allocation algorithm were applied. With a single robot, task allocation reduces to task scheduling. Here, a greedy scheduler with instantaneous task assignment including re-allocation on state changes was used.

Metrics

The primary metric is the number of detected victims. Because in the simulation there are no other victim evidences than heat sources, not the full evaluation sheet as used at RoboCup is applied.

The victims are usually not uniformly distributed in the arena. Additionally, they are typically hidden in corners or at dead ends. This means, that it is possible to explore a large part of the arena, but finding only few victims. Therefore, two different coverage metrics are applied.

The first coverage metric describes the percentage of the complete arena, that is visible in the learned map. This means, the more space was reached by the LRF, the higher is the score. This reflects an internal metric of the robot, because the exploration algorithm is based on this map.

The second coverage metric reflects the percentage of the arena, that has been observed by the thermal camera. Unfortunately, this area can currently not be tracked by the software. However, the robot usually covers its direct surroundings due to the motion pattern of the camera. Therefore, as an alternative to the coverage of the camera, the area covered by the robot’s path is considered. A location is considered as covered, if it is visible from the robot’s path and not further than 1.5 m away from that point.

Experiment Setup

Two different environments were used for the experiments: the layouts of the arenas from German Open 2011 and 2012 (see Figure 7.3). From the potential locations, where typically the victims were placed at the competitions, 5 were picked randomly for the experiments.

Each of the five supervisors ran one trial in each arena. The same number of trials was conducted with a fully autonomous robot. Each trial lasted five minutes.

The interface, which was provided to the supervisors, is depicted in Figure 6.4. It is based on the ROS tools *rviz* and *rqt_gui*. The *rviz* window (Figure 6.4a) displays the map with the position of the robot, including the current facing of the camera. The map is enhanced with overlays of the robot’s past and planned path, and interactive visualizations of all open tasks. The images of the thermal camera (Figure 6.4b) show the same overlays. The *rqt* plugin in Figure 6.4c activates as soon as the robots has found a victim, and queries the supervisor to confirm or discard the victim, or allows to request further inspection of the victim. The mission overview plugin (Figure 6.4d) displays a list of all open tasks, and highlights the task, that is currently allocated to the robot.

The supervisor is enabled to send the following commands to the robot:

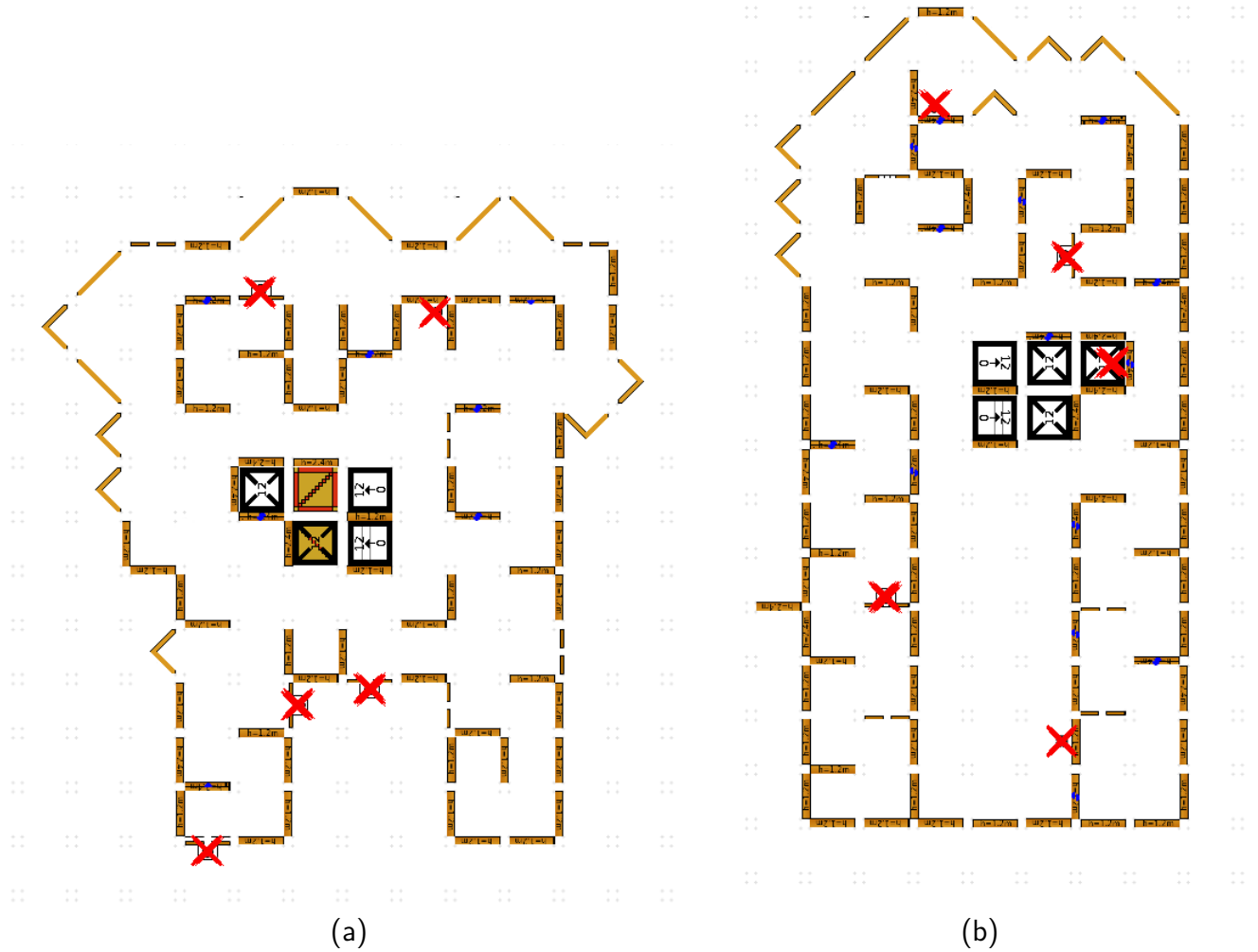


Figure 7.3: (a) Arena layout of German Open 2011. (b) Arena layout of German Open 2012. The victims are marked as red crosses.

- *Add tasks*: A right-click in the map allows to add search tasks or victim tasks to the robot's mission.
- *Delete tasks*: A right click on a task's visualization (Figure 6.4a) allows to delete a task.
- *Change parameters of a task*: A task's position can be changed by dragging the respective marker in the map. The inspection position for a victim hypothesis can be controlled independently of the modeled victim position. Additionally, the supervisor can declare a victim as either true positive (which allows the robot to skip verification of the victim) or false positive (which makes it unnecessary for the robot to further inspect this hypothesis).
- *Request or forbid execution of a task*: This can be achieved either via the menu of the respective task ((Figure 6.4a), or via the tasks overview (Figure 6.4d).
- *Start and stop the robot's autonomy*: The general autonomy mode of the robot can be controlled by a button (Figure 6.4d).

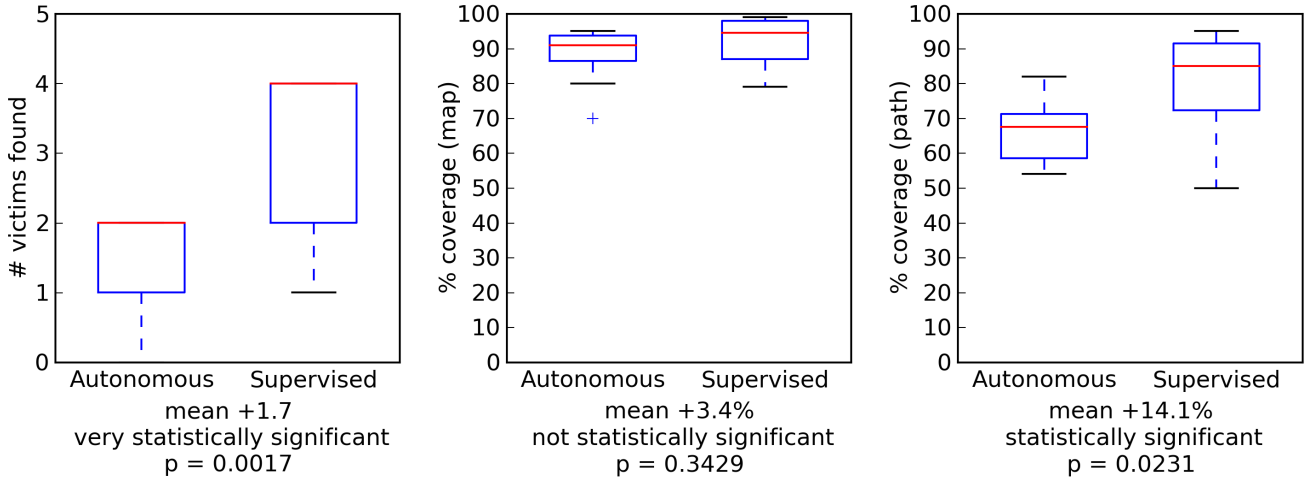


Figure 7.4: Improvements by supervisor support in the USAR mission with instantaneous task assignment.

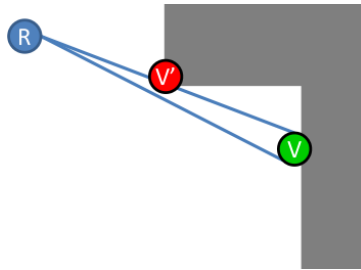


Figure 7.5: Small deviations in the robot's localization or perception can cause large errors in the modeled victim location. In this case, the victim is modeled at the wrong location V' instead of the correct location V.

Results

The results for the 3 applied metrics are depicted in Figure 7.4. The mean of the number of detected victims improved by 1.7 victims, from 1.4 to 3.1. An unpaired t-test showed, that this improvement is very statistically significant ($p = 0.0017$). Regarding the covered area of the LRF map, the mean improved by 3.4%, from 88.3% to 91.7%. However, this is no statistically significant improvement. The area covered by the robot's path improved on average by 14.1%, from 65.6% to 79.7%. This improvement is statistically significant ($p = 0.0231$).

There are different situations, where the supervisors were able to improve the performance of the autonomous robot:

- The autonomous robot discarded several victims because the calculated position to approach the victim was either not reachable, or the victim was not visible for the robot from this position. Similar situations also occurred several times in the supervised missions, however, the participants were able to support the robot in finding the victim by moving the search position to a more appropriate location.
- Wrong projections (as depicted in Figure 7.5) caused the fully autonomous robot to spend more time on some victims as necessary. The supervisors could recognize these situations and either delete the task representing the wrong projection, or drag the victim position to the correct location.

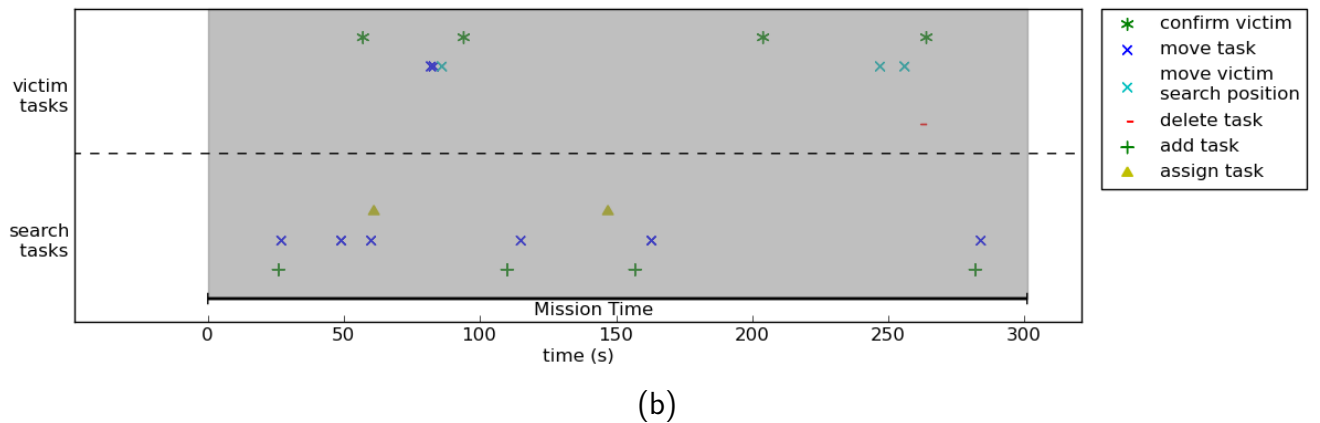
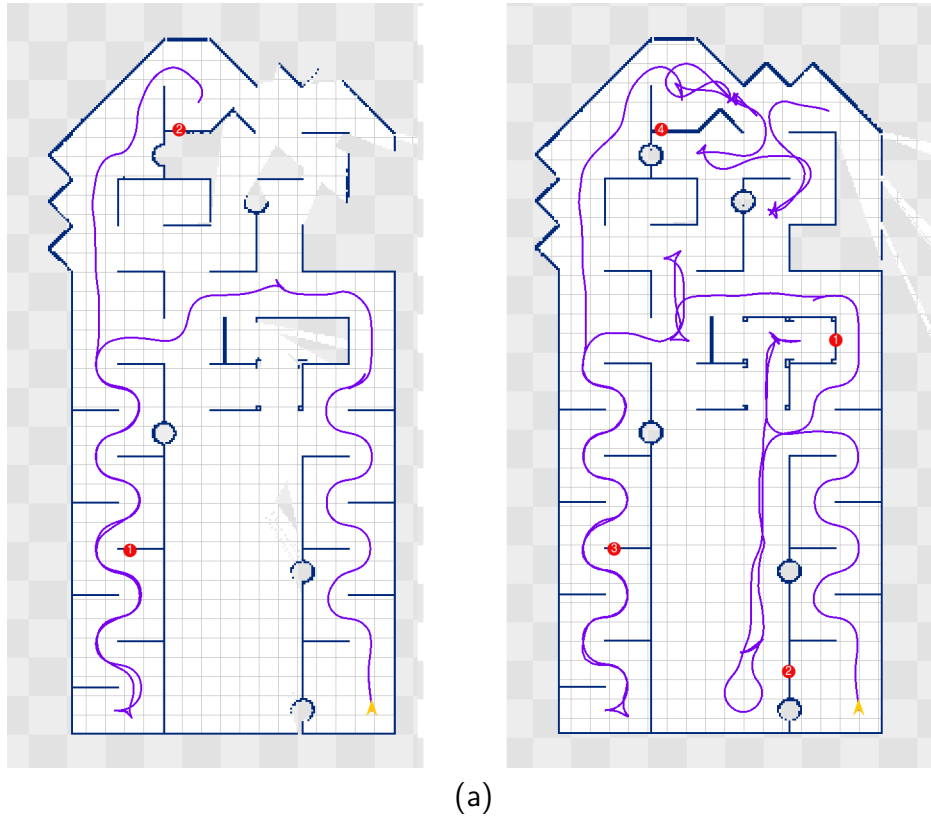
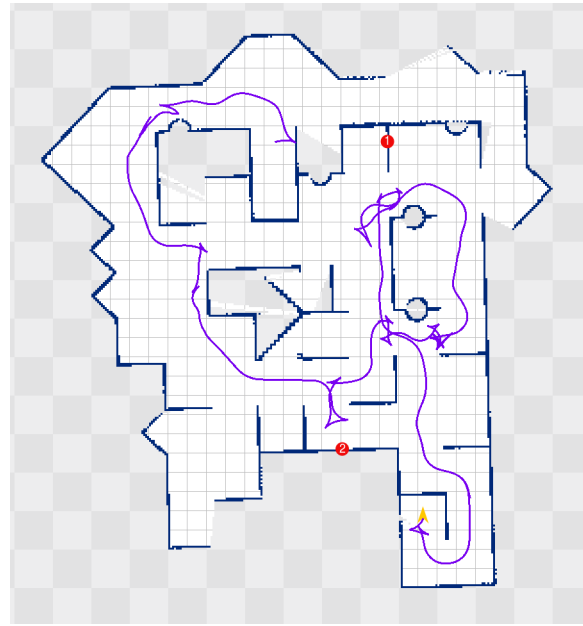
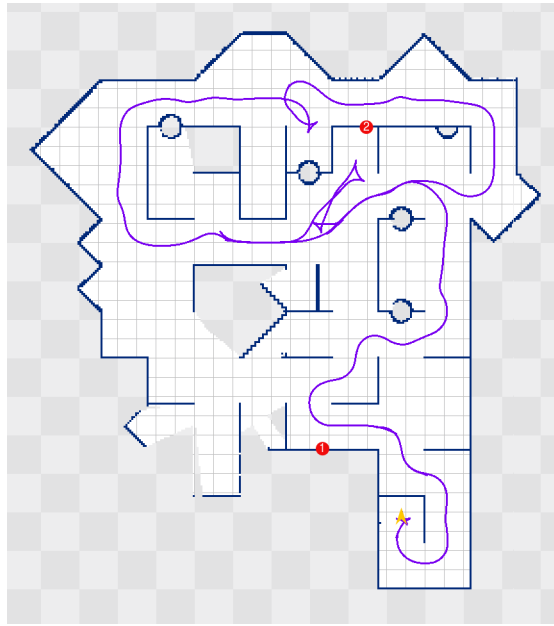
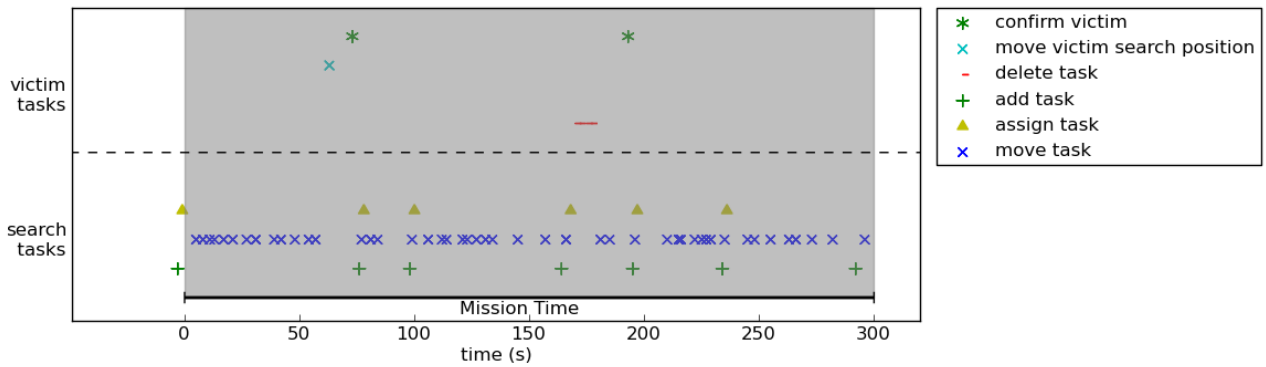


Figure 7.6: Comparison between an autonomous mission and a supervised mission, showing that higher camera coverage leads to more detected victims. (a) Left: a map generated in a fully autonomous run, 93% coverage (map), 58% coverage (path), 2 victims found. Right: a map generated in a supervised run, 98% coverage (map), 92% coverage (path), 4 victims found. (b) Interactions of the supervisor in the run on the right.

- 64% of the victims that were missed by the autonomous robot were located in areas that were not even covered by the LRF map, while 25% of the missed victims were covered by the map, but too far away from the robot's path to be visible to the camera. In the supervised mission, only 13% of the missed victims were caused by a too large distance to the path, while 87% were not detected because they were not even covered by the LRF map. This indicates that in the supervised missions the search in the area covered by the LRF was more exhaustive than in the purely autonomous missions. This effect can also be seen in



(a)



(b)

Figure 7.7: Comparison between an autonomous mission and a supervised mission showing edgy paths due to frequent interactions. (a) Left: a map generated in a fully autonomous run, 90% coverage (map), 72% coverage (path), 2 victims found. Right: a map generated in a supervised run, 90% coverage (map), 72% coverage (path), 2 victims found. (b) Interactions of the supervisor in the run on the right.

Figure 7.6a. The maps show a similar coverage of the LRF map, but the map learned by the supervised robot has a much larger path coverage and more detected victims than the map of the fully autonomous robot.

Figure 7.7a shows, that human input was not in all cases beneficial for the robot's performance. Both runs resulted in a similar coverage of both, path and LRF map, and the same number of detected victims. However, the path of the autonomous robot looks much smoother, because the robot's plans are not interrupted by a human.

When comparing the interactions of supervisor A in Figure 7.6b and supervisor B in Figure 7.7b, it strikes out that supervisor A intervened only sparsely, while supervisor B was almost constantly interacting with the robot. Most of the time, supervisor A trusted in the robot's autonomy, and

only sent commands if the robot was about to leave a large area unexplored, like, for example, the autonomous robot in Figure 7.6a in the large room at the bottom. This shows, that it is beneficial for the supervisor to be familiar with the strengths and weaknesses of the autonomous behavior, because this can lead to less, but more important interactions. Furthermore, interacting less with a single robot gives the supervisor more free time to interact with other robots in a team.

7.4.2 Experiments in Urban Search and Rescue – Time-extended Task Assignment

The experiments refer to the supervision concept with time-extended task assignment, as presented in Chapter 5.3.4, and are submitted for publication in [99].

The inputs from a human supervisor can have positive impacts on both solution quality and computation time. On the one hand, an expert supervisor such as a first responder can have significant implicit knowledge that is not modeled explicitly in the system. For example, the expert knows approximately the whereabouts of people in a collapsed building after a disaster. Therefore, it would make sense to focus the search on this area although this might appear initially as a suboptimal search strategy from an abstract algorithm’s perspective. On the other hand, even basic user inputs in terms of constraints can have substantial impact on the computation time of the algorithm. To solve the MILP in real-time can already for moderate problems turn out

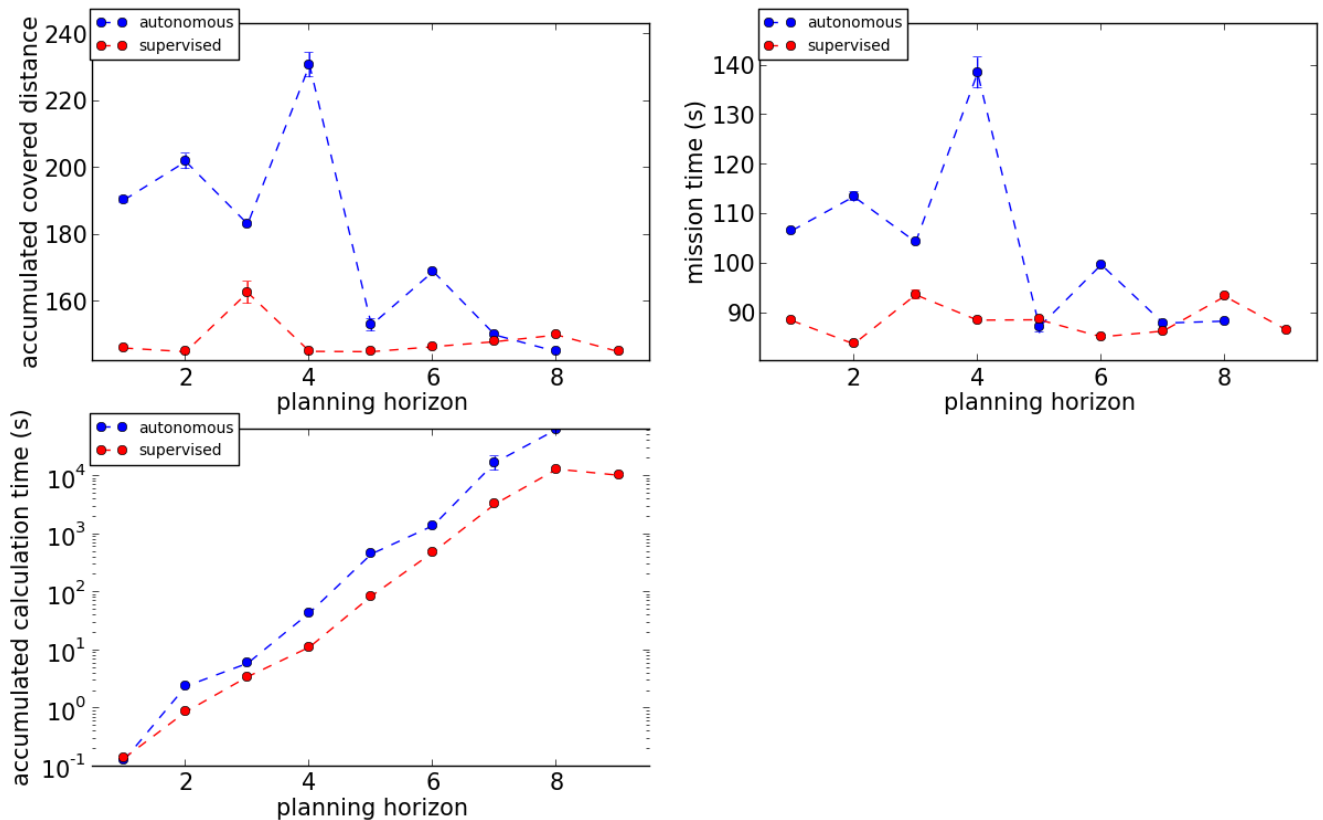


Figure 7.8: Results for the two-rooms example with calculation times depicted in log scale (bottom left figure). Note that results for calculations of planning horizon 9 without human supervision have been omitted due to memory limits on the used computer. Calculated on an Intel Core i7, 4x 3.5GHz, 16GB RAM.

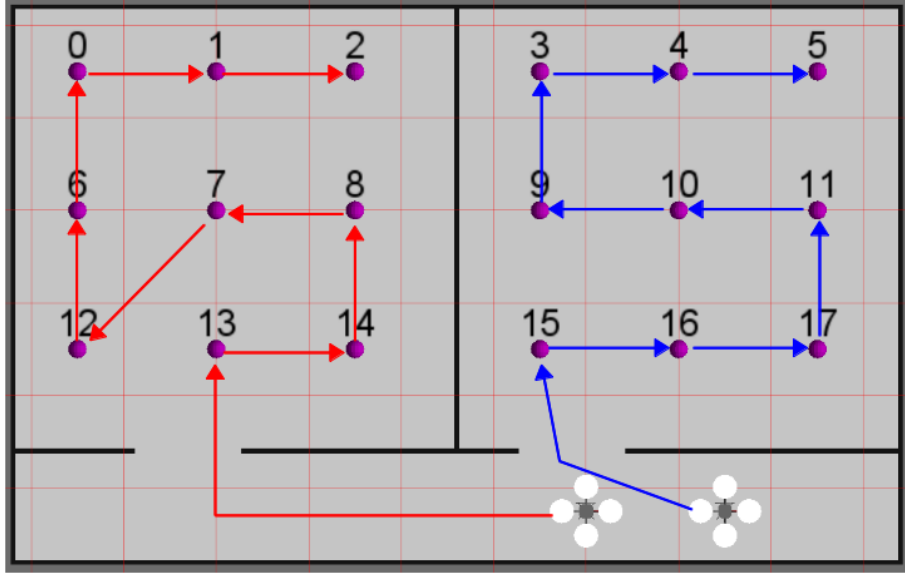


Figure 7.9: Optimal solution for the problem in Figure 5.7 found with planning horizon 9.

to be computationally infeasible. In some scenarios crucial parts of the optimal solution can be immediately apparent to a human supervisor, but need extraordinary time to be computed by a solver. For example, a human can easily identify clusters of tasks that are optimally assigned to a single robot rather than sharing them among the team which would induce additional travel costs.

All implementations in the presented experiments are utilizing ROS (Robot Operating System, [107]). The MILP is modeled using the python interface of Gurobi [5]. The performance is compared between a fully autonomous robot team and a supervised autonomous robot team. All supervisor inputs are given prior to the start of the mission, to avoid biases due to the interface or human performance. Results are presented from both simulated and real-world experiments. The results from simulation are separated into a homogeneous and heterogeneous robot team cooperation scenario.

For user inputs, the graphical user interface *rviz* from the ROS library with *interactive markers* is used for expressing constraints between robots and tasks. Note that this interface is not optimized for efficiency since this is not the primary focus of this work.

Simulation: Homogeneous Cooperation

In the first set of experiments, two unmanned aerial vehicles (UAVs) have to cooperatively explore a small environment consisting of two rooms as shown in Figure 5.7. The problem is solved in an incremental manner by producing sequences of multi-robot team assignments each of them having maximally the length of the pre-defined planning horizon p . Hence, the length of p can have significant influence on the solution quality, particularly if it is smaller than the length needed for the optimal solution.

The solutions computed by the autonomous solver are compared to the solutions computed with human assistance. In this experiment the input by the supervisor was simply to assign for the first optimization step one robot to task 13 in the left room shown in Figure 5.7. Any other assignment was computed autonomously by the solver.

planning horizon	rows		columns		nonzeroes	
	auton.	superv.	auton.	superv.	auton.	superv.
1	2262	2263	940	940	7930	7931
2	2952	2953	978	978	10132	10133
3	3642	3643	1016	1016	12334	12335
4	4332	4333	1054	1054	14536	14537
5	5022	5023	1092	1092	16738	16739
6	5712	5713	1130	1130	18940	18941
7	6402	6403	1168	1168	21142	21143
8	7092	7093	1206	1206	23344	23345
9	7782	7783	1244	1244	25546	25547

Table 7.1: Number of rows (constraints), columns (variables) and nonzeroes (dependency between rows and columns) for the problem in Figure 5.7.

planning horizon	rows (presolved)		columns (presolved)		nonzeroes (presolved)	
	auton.	superv.	auton.	superv.	auton.	superv.
1	1064	1028	126	109	2880	2469
2	2060	1408	810	468	16704	9500
3	2712	2315	846	792	21674	21433
4	3364	3000	882	862	24048	23709
5	4016	3652	918	898	26244	26288
6	4668	4304	954	934	28458	28492
7	5320	4956	990	970	30654	30696
8	5972	5608	1026	1006	32850	32900
9	6624	6260	1062	1042	25364	23989

Table 7.2: Number of rows, columns and nonzeroes for the problem in Figure 5.7 after applying presolve.

The results are summarized in Figure 7.8. 10 trials per configuration were run, except for planning horizon 9 due to memory limits. An optimal solution is visualized in Figure 7.9.

It can be seen that up to a planning horizon of 4 tasks ahead (which is less than half of the tasks per room) the traveled distance and mission time is much higher for the autonomous trials. This is because both robots first accomplished some tasks in the right room, and afterwards both traveled to the left room to work on the remaining tasks. The autonomous missions with planning horizon 4 are exceptionally bad because the robots explored 8 targets in the right room, at this point it was optimal for both robots to continue in the left room, and hence one robot had to return to the first room afterwards. However, for a planning horizon of 5 or higher, it already takes more than 7 minutes to autonomously calculate the solution, compared to less than 90 seconds with a single input from a supervisor. Hence, with input from a supervisor, not only a better solution is found with a smaller planning horizon, also with the same planning horizon the solution is calculated in much shorter time.



Figure 7.10: Real-world experiment with two *Ar.Drone 2.0* quadcopters.

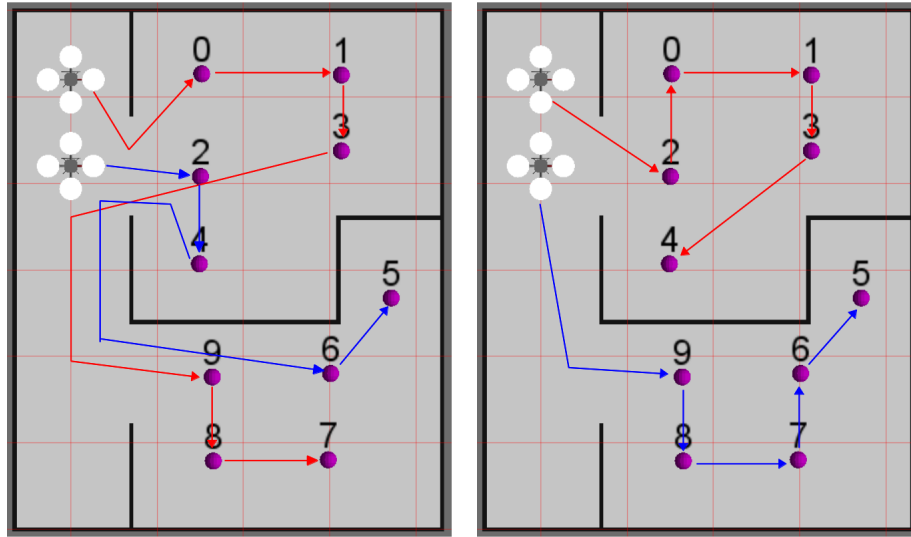


Figure 7.11: Routes of the quadcopters in the real experiments. Left: fully autonomous. Right: manual allocation of task number 9.

Having a look at the size of the MILP, it can be seen in Table 7.1, that the problem size is equal for both, the supervised and the autonomous trials, except for only *one constraint*, which is the one that has been added by the supervisor. As shown in Table 7.2, this constraint is very important since it facilitates a substantial reduction of the problem size when executing the fast presolve algorithm of the MILP solver leading to the speed-up in the subsequent computation.

For the next homogeneous experiment, the supervisor generated two groups, one for targets in the left and one for targets in the right room, that were translated into soft constraints for suggesting joint execution by the same robot. While the optimal sequence of handling tasks had still to be computed by the solver, the input significantly speeds up the computation. For planning horizon 9 the average calculation time was more than 50% lower than for the autonomous calculation with planning horizon 8. (Note that fully autonomous scheduling with planning horizon 9 could not be calculated.) The calculated optimal solution is shown in Figure 7.9.

Real Quadcopter Experiment: Cooperative Exploration

The same experimental setup as described above was used for conducting real-world experiments on two *Ar.Drone 2.0* quadcopters (Figure 7.10). Since the main interest is the coordination algorithm and execution times, a *Vicon Nexus Tracking System* was used composed of ten cameras and reflecting markers attached to the tracked objects for obtaining highly accurate pose estimates of the robots. On top of this, the same ROS-based software was used as in the previous experiments for trajectory planning, combined with a controller based on [31]. Furthermore, the existing ROS AR.Drone driver had to be extended for allowing to control several UAVs at the same time.

As can be expected due to observations from previous simulation runs, a short planning horizon leads to the strategy where both robots are entering the first room together, visiting all targets there, and then approaching the second room. When initially assigning one task in the second room to one of the robots, targets are visited more efficiently, i.e., in both rooms simultaneously. The resulting allocations for both experiments can be seen in Figure 7.11.

However, another important lesson was learned when executing this experiment in the real-world instead of inside the simulation environment: The wind caused by a flying quadcopter can largely influence the flight properties of another quadcopter flying in the close vicinity. This makes it much more challenging for algorithms to control the robots properly, particularly when they are operating close to each other. Therefore, the computed team coordination, that typically tends to distribute the robots, is not only reducing mission time, but also lowers the risk of colliding quadcopters. The video accessible at <http://youtu.be/zojjc2F0fQA> shows the performance of the UAVs with and without supervisor support.

Simulation: Heterogeneous Cooperative Search and Rescue

For experiments with a heterogeneous robot team, the more complex scenario of a large RoboCup rescue arena as used in RoboCup 2009 was considered (Figure 7.12). The arena features different terrain types (flat, ramps, light and heavy stepfields), that require the robots to have different navigation capabilities. The three robots in the team are a wheeled robot, a tracked robot, and a UAV. The wheeled robot can navigate fast on flat floor, but cannot negotiate stairs or stepfields. The tracked robot can negotiate all obstacles except walls, but is very slow. The UAV can fly to all locations very fast, but landing is risky, especially on stepfields, or even impossible, for example on stairs or steep ramps.

Since in this scenario different robot types are inducing different travel costs with respect to the terrain, an efficient path cost planner has been utilized. The travel cost planner is based on value iteration, a popular dynamic programming algorithm frequently used for robot planning [10]. As shown in Figure 7.12 the planner takes as input a classified elevation map in which important structural elements such as stairs and ramps are discriminated. Value Iteration computes efficiently for each grid cell (e_x, e_y) on the elevation map the costs for reaching a goal cell (g_x, g_y) . These costs are composed of travel distance as well as costs for overcoming different types of terrain indicated by the classification.

Three experiments were run with heterogeneous robots in the large arena. The results are summarized in Table 7.3.

For the first experiment in this arena, search tasks were defined in the two flat areas (in the bottom left around the starting location, and in the upper right area) and on the second level, which is reachable via stairs or a steep ramp (Figure 7.13).

A human can recognize quite easily, that it is best to send the UAV to the other side of the arena (tasks 1–9), and let the ground robots work on the tasks close to the starting location (tasks 10–

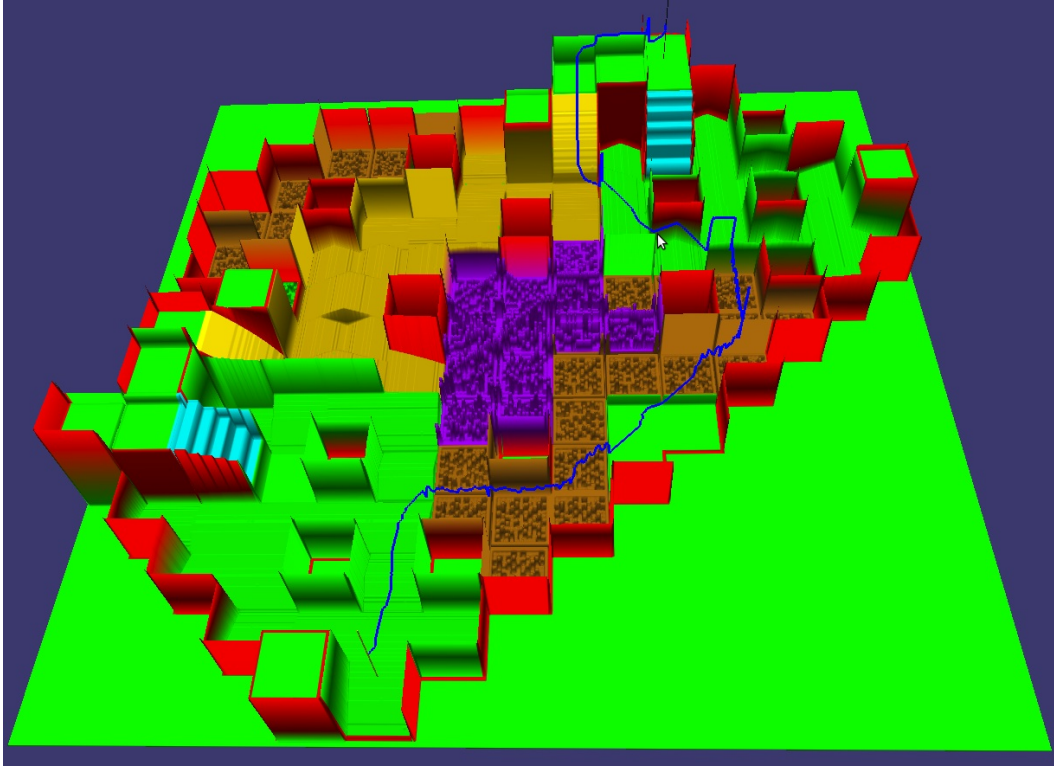


Figure 7.12: Computation of a plan (blue line) and plan costs on a classified elevation map (Model of the RoboCup Rescue arena 2009). Showing traversable (green) and non-traversable (red) terrain. Furthermore, showing stairs (cyan), ramps (yellow), heavy stepfields (violet), and light stepfields (orange) .

20), because the wheeled robot cannot cross the stepfields at all, and the tracked robot needs much more time to negotiate the difficult terrain than the UAV. However, without supervisor input and with a short planning horizon, the UAV works first on tasks close to the starting area (and hence takes away tasks from the UGVs), before flying to the other side of the arena. The supervisor input requires the UAV to have task number 7 in its first schedule. With a greedy scheduler (planning horizon 1), this assignment immediately leads to an improvement of the mission time by 35% compared to the fully autonomous trial. In both cases, the calculation time is very low. In this specific configuration, planning horizon 4 is sufficient for the planner to autonomously send the UAV to the other side of the arena immediately. In that case, the mission time stays the same also with supervisor input, however, the time to calculate this solution is reduced by 65% (almost 3 minutes compared to less than 1 minute) with this single input. This indicates that the results from Section 7.4.2 can be transferred to larger scenarios with heterogeneous team members.

For the second experiment, a timing constraint was added to task number 14. This constraint cannot be met by the tracked vehicle, because driving up the stair or ramp takes too much time. The wheeled robot cannot reach this location at all, hence, the only robot who can meet this constraint is the UAV. The scheduling, especially with the timing constraints for victims, is difficult for a human, but the MILP solver can take care of this. As before, the supervisor requests the UAV to have task number 7 in its first schedule. With fully autonomous scheduling and planning horizon 2, the UAV executes the sequence (17, 14, 11, 10, 4, 2, 1, 3, 5, 8, 9, 6, 7), hence, the robot executes a task on its way to the victim, and also executes tasks on the way to the other side of the stepfields. It turns out that the mission time for both solutions differs only

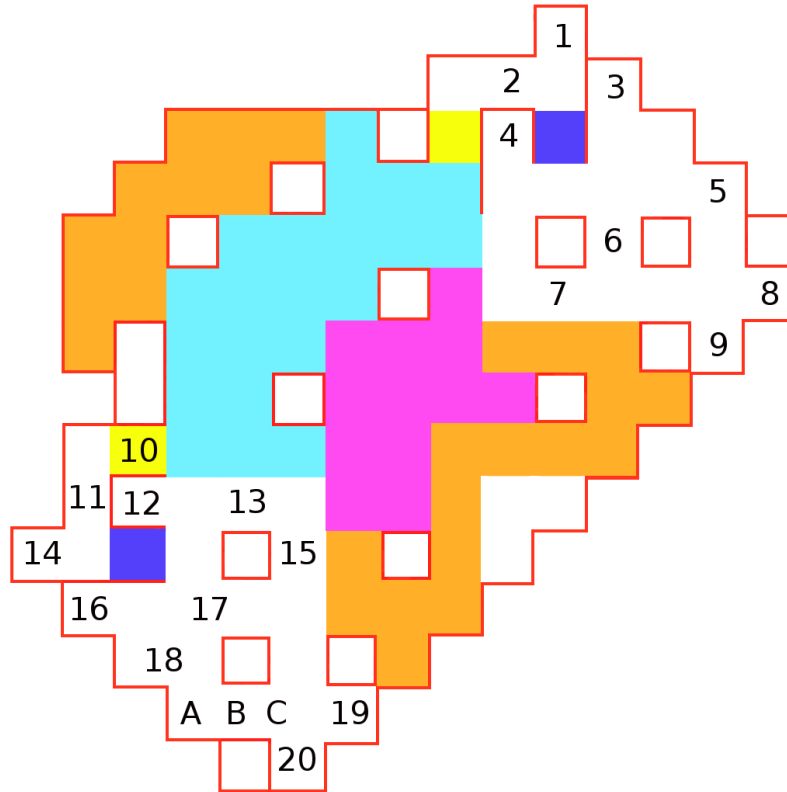


Figure 7.13: Setup for experiment 1 in the large arena. The colors specify different terrain types as in Figure 7.12. Numbers specify exploration tasks. A, B, and C are the starting positions for the three robots.

marginally. However, the supervised solution required only 50% of the calculation time compared to the autonomous solution.

In the third experiment, a further timing constraint was added to task number 3. The tendency of the results are similar as before: the solution quality is similar in both trials, but the timing constraints make the MILP much more difficult to solve. The autonomous planning takes 45 seconds, while the planning with the single human input is reduced to 12 seconds.

7.4.3 Speed-up Optimization using Queries

Consider again the experiment setup of the previous section. In this experiment, the solver always calculated until the optimal solution was found. However, often an incumbent (the currently best detected solution) is already good enough to be executed, so that there is no need of waiting until the end of the optimization.

It is possible to automatically stop the optimization as soon as a predefined performance bound is reached. However, for the given problem the major part of the objective value is defined by the revenue earned for completing the tasks. Therefore, the solver quickly finds a solution with an objective value that is known to differ less than 2% from the optimal solution, but for a human this solution looks much worse than the optimum. Furthermore, changing the revenue values affects where this bound must be chosen, to end up with the same result. Hence, automatic interruption of the optimization is very dangerous and can lead to undesired behavior.

Time constraints	Mode	Planning horizon	Mission time	Travel costs	Calc. time
none	autonomous	1	1160.4	2366	0.4
	supervised	1	855.8	1922	0.4
	autonomous	4	826.2	1863	159.8
	supervised	4	843.6	1874	57.7
task 14	autonomous	2	982.8	2370	9.7
	supervised	2	1009.4	2380	4.8
tasks 3, 14	autonomous	2	1096.2	5510	45.3
	supervised	2	962.6	5603	11.8

Table 7.3: Results for supervision experiments with heterogeneous robots in the RoboCup rescue arena. Calculated on an Intel Xeon W3565, 4x 3.2GHz, 16GB RAM.

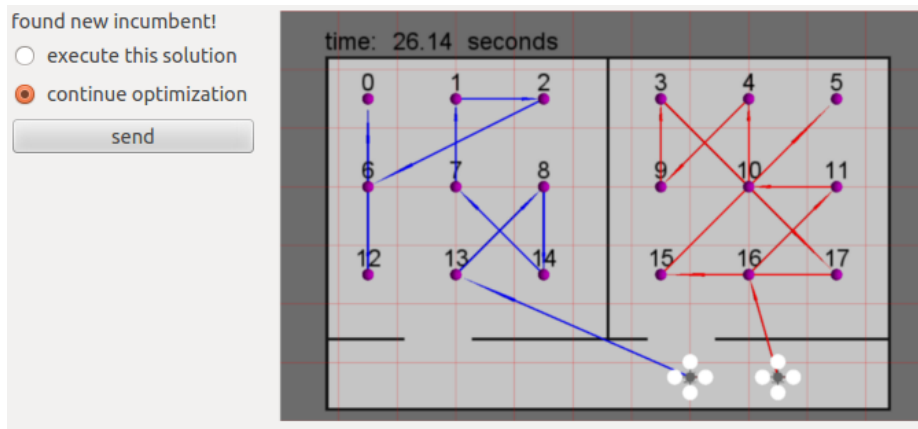


Figure 7.14: Example of a query showing the current incumbent.

To leave the decision whether a solution is good enough or not to the supervisor, a query is generated every time the solver finds a new incumbent (Figure 7.14). The supervisor can decide to (a) stop the optimization and execute the current solution, or (b) continue with the optimization to find a better solution. This query is sent as an AV query (autonomous with veto), with the default value set to answer (b), hence the solver continues until it is interrupted by the supervisor with answer (a).

The incumbents are attached to the query as image payload and can be seen in Figure 7.15 for this example. In the first three feasible solutions, the robots switch between the two rooms, and therefore these solutions are not good enough, on the one hand because execution of this solution takes too long, and on the other hand because the risk of colliding robots is very high with these solutions. Already after less than 30 seconds, the structure of separating the two drones to the two rooms is found, but the paths of both robots are very disordered. However, only 5 seconds later a solution is found that looks much cleaner. This is more than 10 seconds faster than the average calculation time for planning horizon 4 (with fully autonomous solving), but the solution quality is much better, because none of the robots has to switch rooms. After 209 and 258 seconds of calculation time, solutions are found that are very close to the optimum. This is still twice as fast as the fully autonomously calculated solution for planning horizon 5 (the minimum planning horizon for ending up with the optimal structure). Finally, after almost 2800 seconds of calculation time, an optimal solution is found. This means, that the solver, if not interrupted, spends more

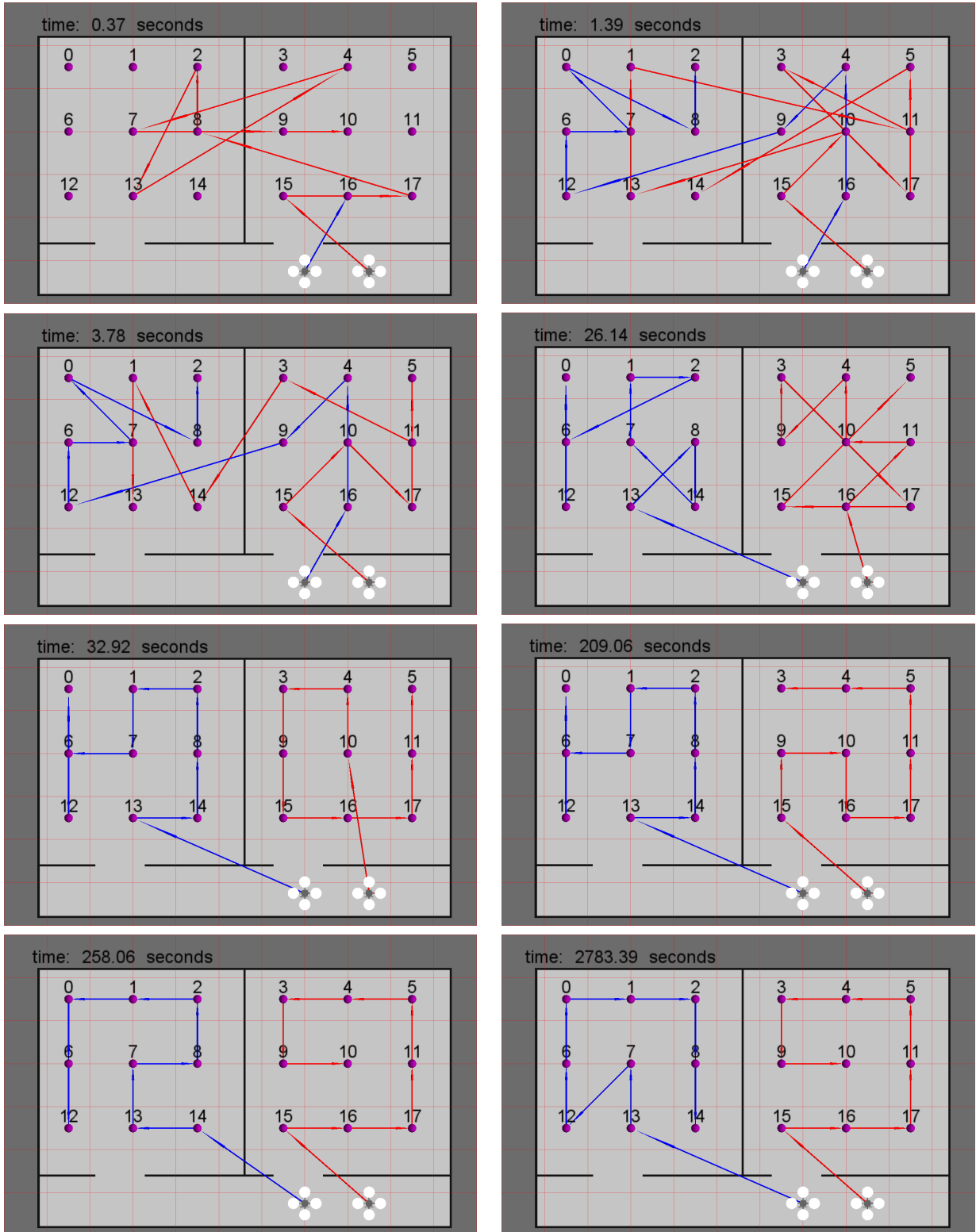


Figure 7.15: Incumbents of the optimization for the two rooms example. After less than 30 seconds, the solution structure of distributing the two robots on the two rooms is found. A very good solution is found after 33 seconds, and a solution very close to the optimum is found after 210 seconds. The optimum is found after 2800 seconds, the autonomously calculated solution for planning horizon 8 took more than 23 times the amount of calculation time.

Tactic	Role 1	Role 2	Role 3	Role 4
defensive	striker	goalkeeper	libero	supporter
medium	striker	goalkeeper	supporter	supporter
offensive	striker	goalkeeper	assist striker	supporter
very offensive	striker	assist striker	supporter	supporter

Table 7.4: Different tactics used in soccer matches.

than 20 times the amount of calculation time to find out that this is actually the best solution. These results were obtained on the same computer (Intel Core i7, 4x 3.5GHz, 16GB RAM) as the experiments without queries.

For most supervisors, the solution found after 33 seconds of calculation time is good enough, and hence they would decide to stop the optimization at this point. However, some other supervisor might want to wait for an even better solution, and interrupt the optimization at a later point in time. Hence, queries are a good means for enabling a human supervisor to interact with the solver in a flexible manner.

7.4.4 Application in Robot Soccer - Instantaneous Task Assignment

In robot soccer, no human interactions during a match are allowed by the rules. Nevertheless, supervision commands can be used to facilitate the behavior development process and to improve the team behavior in a soccer match. These two applications are described below, after a short overview about the used approach to tactics and role allocation.

Background: Role Allocation and Tactics in Robot Soccer

The team behavior of the Darmstadt Dribblers is based on different roles, that define the behavior of a robot. These roles are described in Chapter 2.1.1. Each role features some parameters, that allow to change details of the behavior within the specific role.

A tactic describes a collection of different roles to achieve a common team behavior. A striker is always part of each tactic, all other roles are optional, and can also occur more than once, but should have a different parametrization for each occurrence. The roles within a tactic are sorted by priority, which is used to allocate the roles to the robots (see below). Each tactic consists of one role more than players in the team, to have a fallback if one role cannot be executed by any robot, for example, if no suitable goalkeeper robot is available. Some example tactics are listed in Table 7.4.

During a match, the robots have to negotiate which robot executes which role within the current tactic. Because the situation on the playing field changes very fast, a highly responsive instantaneous role assignment is used.

Every robot calculates the costs for executing each available role, dependent on the robot's own current position and the position of the ball. The costs reflects the estimated time for the robot to reach the desired target position of the respective role, for example a position close to the ball for the striker role. To prevent frequent oscillations between roles, each robot multiplies the costs for its current role with a factor $\alpha < 1$ to achieve a hysteresis. Furthermore, the robots have to keep their role for a minimum time $\beta > 0$ ms. During that time, the costs for the own role are set to 0 and for all other roles to ∞ .

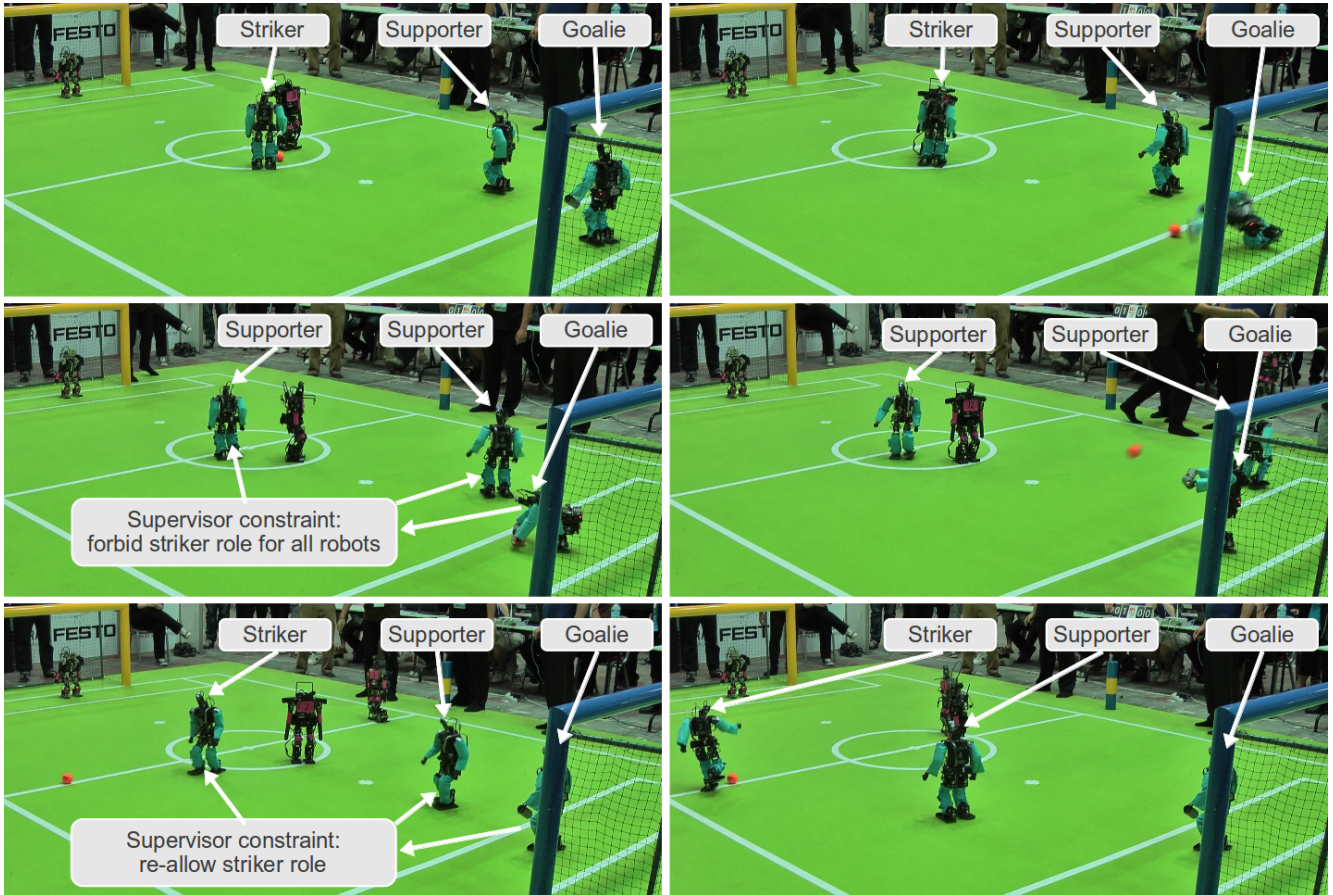


Figure 7.16: A scene from the quarter final at RoboCup 2011, showing the clearing action of the goalkeeper. While the goalkeeper clears the ball, the two field players are not allowed to execute the striker role, because of an artificial supervisor constraint, and therefore stay away from the ball. After the clearing action is finished, the supervisor constraint is canceled, and one of the field players approaches the ball again.

The allocation of roles to robots is *not* desired to be globally optimal, the roles are rather allocated in the order of their priority, because it is more important to control the ball as fast as possible, than to reach a good supporter position. Because of unreliable communication, and for the sake of a high responsiveness, the robots do not negotiate the roles explicitly, instead, every robot executes the same deterministic algorithm to determine its own role, based on own calculated costs and the costs received from the teammates. The role with the highest priority is assigned to the robot with the lowest costs for this role. Afterwards, the role with the second highest priority is assigned to the best suited of the remaining robots and so forth, until all robots are assigned. In that way, no explicit negotiation is necessary. Due to communication delays short periods of inconsistent allocations are possible, but these are short enough to be less critical than explicit negotiations between all team members. In Figure 7.17, such a short period of inconsistent allocation is resolved within less than 0.2 seconds.

Tactics Refinement

Although human interactions are not allowed during a soccer match, methods from Chapter 5.3 can be used to refine tactics during test matches or in simulation. A supervisor can change the role types involved in the current tactic, adapt the role priorities, or modify the parameters for

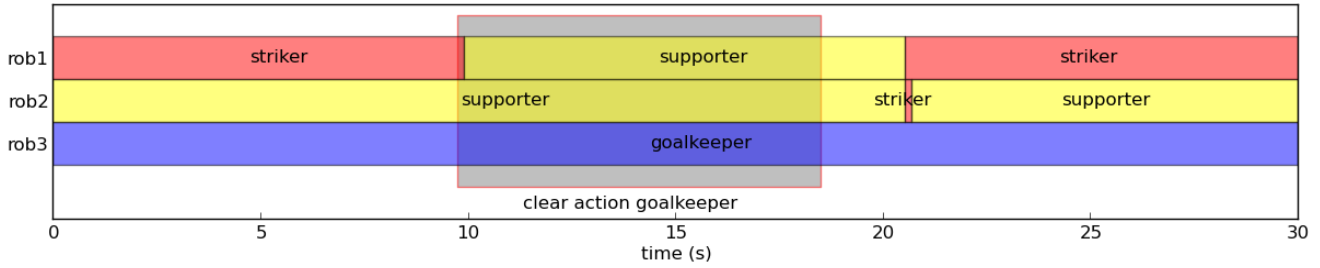


Figure 7.17: Visualization of the role allocation for the simulated clearing action of the goalkeeper.

each role. If a simulation with the expected opponent behavior is available, it is even possible to automatically optimize the parameters for a given tactic. However, this requires extensive simulations, because for each parameter set at least one match needs to be simulated.

In addition to the common tactic parameters, also individual robot capabilities can be evaluated prior to a match, which are then translated into supervisor constraints during a match. If, for example, it is known that one of the robots cannot kick reliably due to bad calibration or wear, it should defer the striker role (i.e., multiply its costs for executing the striker role by a factor $a > 1$). In that way, if two robots are in the same distance to reach the ball, the robot with the better kicking abilities will execute the striker role, while the other robot acts as backup, and only needs to kick if all other robots are in much worse positions.

Autonomous Supervision Agent

An important application of supervisor commands during a soccer match is the modeling of exceptions from the role allocation algorithm defined above. These exceptions are realized using artificial supervision commands, that the robots generate autonomously, as reactions to predefined situations within a match.

By the rules, there must be one dedicated goalkeeper robot. This is achieved with a fixed allocation of this robot to the goalkeeper role. In terms of costs, this robot's costs for executing the goalkeeper role is set to zero, and for all other roles to infinity. Furthermore, all other robots are not allowed to execute the goalkeeper role, and hence have infinite costs for this role. In case the goalkeeper is the last robot on the field (e.g., all other robots are penalized), the fixed allocation is canceled, which leads to an allocation of the goalkeeper robot to the role with the highest priority (usually the striker role). If another robot is back in the game, the goalkeeper robot waits until the other robot is in a better position for the striker role, and then re-activates the fixed allocation to the goalkeeper role.

If the ball is close to the own goal, and the goalkeeper decides to clear the ball, it sends a constraint to all other robots, that forbids to execute the striker role. In that way, the goalkeeper has exclusive control over the ball. As soon as the clear action is finished, the goalkeeper cancels this constraint. Figure 7.16 and the video at <http://youtu.be/M82pM9VV01k> show such a situation in the quarter final from RoboCup 2011. Figure 7.17 shows the exact timings for a similar situation, that was produced artificially in simulation. The corresponding video can be seen at http://youtu.be/_BQiwWf4ItI

8 Conclusion

The presented concepts for supervision of autonomous robot teams can be applied to a variety of different problem classes to significantly improve progress in mission achievement by utilizing complementary abilities of humans and robots. The application to urban search and rescue and robot soccer has been shown in experiments, and further examples were discussed for related problems as the DARPA robotics challenge and a production hall logistics example. The presented applications vary in their criticality, heterogeneity of the involved robots, and structuring of the environment. They have in common that a single human supervisor interacts with a team of autonomous robots from a remote location. The developed concepts can also be transferred to other applications with different characteristics.

Both, the capabilities of humans and robots, as well as the way humans work together in similar situations indicate, that the supervisor role has a high potential of enhancing the performance of an autonomous robot team. Therefore, the presented concepts are in many aspects inspired by the human teamwork models.

As an appropriate knowledge base for a human supervisor, the new concept of situation overview has been developed. Other state of the art awareness concepts turned out to be not appropriate for the role of the supervisor. Awareness concepts for one-to-one interactions between a human and a robot include lots of details that are not relevant for the targeted interactions, and also would overburden the human when trying to obtain the same knowledge for a whole team of robots. Awareness concepts for robot teams disregard aspects related to individual members of the team, like each robot's health and individual performance, and therefore lack important information. The developed concept of situation overview (SO) addresses both, an overview about team coordination and mission progress, as well as broad information about each robot's current state and actions.

For obtaining SO, an event-based communication system has been presented. It allows the robots to detect events that are relevant for the supervisor's SO using methods from complex event processing. Tags based on semantic topics and criticality levels allow on the one hand different presentation of the events at a user interface, and are on the other hand a basis for policy-based filtering of the events before sending. This communication concept is inspired by loosely coupled human teamwork and requires a low communication overhead compared to standard teleoperation methods, because only data needed for situation overview are sent, while details that are only used for exact teleoperation are omitted. The methods enable a human supervisor to gain a general situation overview of a whole robot team, without requiring the supervisor to be familiar with implementation details.

Based on SO, the supervisor can interact with the robot team. Interactions initiated by the robots are modeled as queries, that allow the robots to transfer decisions to the supervisor. Many of those decisions are difficult to take for an autonomous algorithm, but comparably easy for a human, who can base decisions on SO and expertise, which reflects the discussed superiorities of humans over robots. Different query modes, ranging from fully autonomous decisions to pure human decisions, allow to dynamically adapt the robots' level of autonomy.

Interaction initiated by the supervisor are separated into two categories: on the one hand commands that change the set of tasks that compose the mission, and on the other hand commands that affect the allocation of tasks to robots. With instantaneous task assignment, these commands

are realized by only modifying the costs for executing the tasks. Therefore, the developed method can be applied to different task allocation methods, for example centralized, behavioral, or market based approaches. For time-extended task assignment, a novel MILP formulation is presented. Here, the intuitive commands from the human supervisor are internally translated into MILP constraints. This enables the supervisor to define parts of the solution, which supports the solver in finding a solution faster than without any support. Furthermore, the commands from the human can cut off locally optimal solutions, which is especially important for incremental solving with limited planning horizon, because the structure of the globally optimal solution can be entirely different from local optima. Experiments showed that with both methods already few inputs by a human can significantly improve the solution quality, e.g., regarding the number of detected victims, the mission duration, and the required calculation time in urban search and rescue scenarios. Timing constraints on some tasks, that are not easy to understand for a human, are handled well by the robots autonomously.

Overall, the presented methods provide an extension to the autonomy of a robot team, that allow to integrate a human supervisor, while preserving all advantages of an autonomous team. The methods are independent of the scenario, and can be applied to fundamentally different problem classes, featuring structured or unstructured environments, homogeneous and heterogeneous robot teams, and a high degree of robot autonomy. The methods make use of the complementary abilities of robots and humans. While robots can achieve tasks autonomously, that often cannot be teleoperated well or would overburden a single human for a whole team, a human supervisor can compensate the weaknesses of autonomous robots, by incorporating external knowledge and expertise. Hence, with the methods developed in this thesis, supervision of autonomous robot teams can significantly improve the team's performance.

Bibliography

- [1] Feuerwehr sucht mit Robotern nach Überlebenden. Welt Online, March 2009.
 - [2] Kölner Feuerwehr hat keinerlei Hinweise auf Vermisste. Spiegel Online, March 2009.
 - [3] Team Darmstadt Dribblers website: <http://www.dribblers.de/>, 2013.
 - [4] Team Hector Darmstadt website: <http://www.gkmm.tu-darmstadt.de/rescue/>, 2013.
 - [5] Gurobi optimizer reference manual, 2014. Gurobi Optimization, Inc., <http://www.gurobi.com/>.
 - [6] M. Andriluka, P. Schnitzspan, J. Meyer, S. Kohlbrecher, K. Petersen, O. von Stryk, S. Roth, and B. Schiele. Vision based victim detection from unmanned aerial vehicles. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1740 – 1747, October 18 - 22 2010.
 - [7] T. Balch. Social entropy: A new metric for learning multi-robot teams. In *Proceedings of the 10th International FLAIRS Conference*, pages 272–277, 1997.
 - [8] J. M. Bradshaw, H. Jung, S. Kulkarni, M. Johnson, P. Feltovich, J. Allen, L. Bunch, N. Chambers, L. Galescu, R. Jeffers, N. Suri, W. taysom, and A. Uszok. Kaa: Policy-based eplorations of a richer model for adjustable autonomy. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, July 2005.
 - [9] H. Branding, A. P. Buchmann, T. Kudrass, and J. Zimmermann. Rules in an open system: the REACH Rule System. In *Proceedings of the 1st international Workshop on Rules in Database Systems*, pages 111 – 126, Edinburgh, Scotland, September 1993.
 - [10] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. The interactive museum tour-guide robot. In *Proceedings of the National Conference on Artificial Intelligence*, pages 11–18. John Wiley & Sons Ltd., 1998.
 - [11] D. Calisi, D. Nardi, K. Ohno, and S. Tadokoro. A semi-autonomous tracked robot system for rescue missions. In *SICE Annual Conference, 2008*, pages 2066–2069, 2008.
 - [12] J. Casper and R. R. Murphy. Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 33(3):367 – 385, june 2003.
 - [13] S. Chakravarthy and D. Mishra. Snoop: an expressive event specification language for active databases. *IEEE Data and Knowledge Engineering*, 14(10):1 – 26, 1994.
 - [14] J. Y. Chen, M. J. Barnes, and C. Kenny. Effects of unreliable automation and individual differences on supervisory control of multiple ground robots. In *Proceedings of the 6th international conference on Human-robot interaction, HRI '11*, pages 371–378, New York, NY, USA, 2011. ACM.
 - [15] J. Y. Chen, M. J. Barnes, and Z. Qu. Roboleader: an agent for supervisory control of multiple robots. In *Proceeding of the 5th ACM/IEEE international conference on Human-robot interaction, HRI '10*, pages 81–82, New York, NY, USA, 2010. ACM.
-

-
- [16] J. Y. C. Chen, M. J. Barnes, and M. Harper-Sciari. Supervisory control of multiple robots: Human-performance issues and user-interface design. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 41(4):435 – 454, 2011.
- [17] J. W. Crandall and M. L. Cummings. Developing performance metrics for the supervisory control of multiple robots. In *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 33–4. ACM, 2007.
- [18] M. L. Cummings, S. Bruni, S. Mercier, and P. J. Mitchell. Automation architecture for single operator-multiple uav command and control. *International Command and Control Journal*, 1(2):1 – 24, 2007.
- [19] S. da Costa Botelho and R. Alami. M+ : a scheme for multi-robot cooperation through negotiated task allocation and achievement. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1234–1239, Detroit, Michigan, May 1999.
- [20] U. Dayal, A. Buchmann, and D. McCarthy. Rules are objects too: A knowledge model for an active, object-oriented database system. In K. Dittrich, editor, *Advances in Object-Oriented Database Systems*, volume 334 of *Lecture Notes in Computer Science*, pages 129–143. Springer Berlin / Heidelberg, 1988.
- [21] M. B. Dias and A. Stentz. A free market architecture for distributed control of a multirobot system. In *6th International Conference on Intelligent Autonomous Systems (IAS-6)*, pages 115–122, July 2000.
- [22] M. B. Dias and A. Stentz. A comparative study between centralized, market-based, and behavioral multirobot coordination approaches. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '03)*, volume 3, pages 2279–2284, October 2003.
- [23] M. B. Dias and A. T. Stentz. Opportunistic optimization for market-based multirobot control. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '02)*, volume 3, pages 2714 – 2720, September 2002.
- [24] O. Diaz, N. Paton, and p. Gray. Rule mangement in object oriented databases: A uniform approach. In *Proceedings of the 17th International Conference on VLDB*, pages 317–326, Barcelona, Spain, September 1991.
- [25] P. Dourish and V. Bellotti. Awareness and coordination in shared workspaces. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, pages 107–114, New York, NY, USA, 1992. ACM.
- [26] J. L. Drury, J. Scholtz, and H. A. Yanco. Awareness in human-robot interactions. In *IEEE International Conference on Systems, Man and Cybernetics, 2003.*, volume 1, pages 912 – 918, 2003.
- [27] C. A. Ellis, S. J. Gibbs, and G. Rein. Groupware: Some issues and experiences. *Communications of the ACM*, 34(1):39–58, January 1991.
- [28] M. R. Endsley. Situation awareness global assessment technique (sagat). In *Proceedings of the IEEE 1988 National Aerospace and Electronics Conference*, volume 3, pages 789 – 795, 1988.

-
- [29] M. R. Endsley. A methodology for the objective measurement of pilot situation awareness. In *Situational Awareness in Aerospace Operations (AGARD-CP-478)*, pages 1/1 – 1/9, Neuilly sur Seine, France, 1990. NATO - AGARD.
- [30] M. R. Endsley. Toward a theory of situation awareness in dynamic systems. *Human factors*, 37(1):32–64, 1995.
- [31] J. Engel, J. Sturm, and D. Cremers. Camera-based navigation of a low-cost quadcopter. In *Proceedings of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.
- [32] A. Farinelli, L. Iocchi, and D. Nardi. An analysis of coordination in multi-robot systems. In *IEEE International Conference on Systems, Man and Cybernetics, 2003*, volume 2, pages 1487–1492, 2003.
- [33] A. Farinelli, L. Iocchi, D. Nardi, and V. A. Ziparo. Assignment of dynamically perceived tasks by token passing in multirobot systems. *Proceedings of the IEEE*, 94(7):1271–1288, 2006.
- [34] D. H. Fernald and C. W. Duclos. Enhance your team-based qualitative research. *Annals of Family Medicine*, 3(4):360 – 364, July/August 2005.
- [35] D. A. Few, D. J. Bruemmer, and M. C. Walton. Improved human-robot teaming through facilitated initiative. In *The 15th IEEE International Symposium on Robot and Human Interactive Communication, ROMAN 2006*, pages 171–176, 6-8 Sept. 2006.
- [36] P. M. Fitts, editor. *Human engineering for an effective air-navigation and traffic control system*. Washington, DC: National Academy of Sciences Archives, 1951.
- [37] T. Fong, I. Nourbakhsh, and K. Dautenhahn. A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42(3-4):143 – 166, 2003.
- [38] T. Fong, C. Thorpe, and C. Baur. Robot, asker of questions. *Robotics and Autonomous Systems*, 42:235 – 243, 2003.
- [39] A. Franchi, C. Secchi, M. Ryll, H. H. Bühlhoff, and P. R. Giordano. Shared control: Balancing autonomy and human assistance with a group of quadrotor uavs. *IEEE Robotics and Automation Magazine, Special Issue on Aerial Robotics and the Quadrotor Platform*, 19:57–68, 09/2012 2012.
- [40] M. Friedmann. *Simulation of Autonomous Robot Teams With Adaptable Levels of Abstraction*. PhD thesis, Technische Universität Darmstadt, Nov. 30 2009.
- [41] D. Gale. *The theory of linear economic models*. McGraw-Hill, New York, 1960.
- [42] S. Gatzui and K. R. Dittrich. Events in an active object-oriented database system. In *Proceedings of the 1st Workshop on Rules in Database Systems*, Edinburg, August 1993.
- [43] B. P. Gerkey and M. J. Mataric. Sold!: Auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, October 2002.
- [44] B. P. Gerkey and M. J. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.

-
- [45] D. F. Glas, T. Kanda, H. Ishiguro, and N. Hagita. Simultaneous teleoperation of multiple social robots. In *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, HRI '08, pages 311–318, New York, NY, USA, 2008. ACM.
- [46] M. A. Goodrich, T. W. McLain, J. D. Anderson, J. Sun, and J. W. Crandall. Managing autonomy in robot teams: observations from four experiments. In *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 25–32. ACM, 2007.
- [47] M. A. Goodrich and D. R. Olsen, Jr. Seven principles of efficient human robot interaction. In *IEEE International Conference on Systems, Man and Cybernetics, 2003*, volume 4, pages 3942–3948, October 2003.
- [48] M. A. Goodrich, D. R. Olsen, Jr., J. W. Crandall, and T. J. Palmer. Experiments in adjustable autonomy. In *Proceedings of the IJCAI Workshop on Autonomy, Delegation and Control: Interacting with Intelligent Agents*, pages 1624–1629, 2001.
- [49] E. Guizzo. Fukushima robot operator writes tell-all blog. *IEEE Spectrum*, August 2011.
- [50] E. Guizzo. Japan earthquake: Robots help search for survivors. *IEEE Spectrum*, March 2011.
- [51] E. Guizzo. Robotic aerial vehicle captures dramatic footage of fukushima reactors. *IEEE Spectrum*, April 2011.
- [52] E. Guizzo. Robots enter fukushima reactors, detect high radiation. *IEEE Spectrum*, April 2011.
- [53] C. Gutwin, S. Greenberg, and R. Mark. Workspace awareness in real-time distributed groupware: Framework, widgets, and evaluation. In *Proceedings of HCI on People and Computers XI*, pages 281–298, 1996.
- [54] B. Hardin and M. A. Goodrich. On using mixed-initiative control: a perspective for managing large-scale robotic teams. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 165–172, 2009.
- [55] S. G. Hart and L. E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In P. A. Hancock and N. Meshkati, editors, *Human Mental Workload*, chapter 7, pages 139–183. Elsevier, 1988.
- [56] K. Hatazaki, M. Konyo, K. Isaki, S. Tadokoro, and F. Takemura. Active scope camera for urban search and rescue. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2007*, pages 2596–2602, nov 2007.
- [57] A. Hinze, K. Sachs, and A. Buchmann. Event-based applications and enabling technologies. In *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, 1:1–1:15, July 2009.
- [58] A. Jacoff, E. Messina, B. Weiss, S. Tadokoro, and Y. Nakagawa. Test arenas and performance metrics for urban search and rescue robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, volume 4, pages 3396 – 3403, oct. 2003.

-
- [59] A. S. Jacoff, R. K. Sheh, A. M. Virts, T. Kimura, J. Pellenz, S. Schwertfeger, and J. Suthakorn. Using competitions to advance the development of standard test methods for response robots. In *Proceedings of the Workshop on Performance Metrics for Intelligent Systems*, pages 182–189, 2012.
- [60] M. Johnson, J. M. Bradshaw, P. J. Feltovich, C. M. Jonker, B. Riemsdijk, and M. Sierhuis. The fundamental principle of coactive design: Interdependence must shape autonomy. In M. Vos, N. Fornara, J. Pitt, and G. Vouros, editors, *Coordination, Organizations, Institutions, and Norms in Agent Systems VI*, volume 6541 of *Lecture Notes in Computer Science*, pages 172–191. Springer Berlin Heidelberg, 2011.
- [61] M. Johnson, P. J. Feltovich, J. M. Bradshaw, and L. Bunch. Human-robot coordination through dynamic regulation. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, pages 2159 – 2164, Pasadena, CA, May 19 - 23 2008.
- [62] E. G. Jones, B. Browning, M. B. Dias, B. Argall, M. Veloso, and A. Stentz. Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, Orlando, Florida, May 2006.
- [63] M. W. Kadous, R. K.-M. Sheh, and C. Sammut. Controlling heterogeneous semi-autonomous rescue robot teams. In *Proceedings of the 2006 IEEE International Conference on Systems, Man, and Cybernetics*, volume 4, pages 3204 – 3209, 2006.
- [64] M. W. Kadous, R. K.-M. Sheh, and C. Sammut. Effective user interface design for rescue robotics. In *HRI '06: Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 250–257. ACM, 2006.
- [65] N. Kalra, D. Ferguson, and A. Stentz. Hoplitest: A market-based framework for planned tight coordination in multirobot teams. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1170–1177, Barcelona, Spain, April 2005.
- [66] B. Kalyanasundaram and K. Pruhs. Online weighted matching. *Journal of Algorithms*, 14(3):478–488, 1993.
- [67] T. Kaupp and A. Makarenko. Measuring human-robot team effectiveness to determine an appropriate autonomy level. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, pages 2146 – 2151, Pasadena, CA, USA, May 19 - 23 2008.
- [68] G. Klein, D. D. Woods, J. M. Bradshaw, R. R. Hoffman, and P. J. Feltovich. Ten challenges for making automation a “team player” in joint human-agent activity. *Intelligent Systems, IEEE*, 19(6):91 – 95, nov.-dec. 2004.
- [69] M. Koes, I. Nourbakhsh, and K. Sycara. Constraint optimization coordination architecture for search and rescue robotics. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3977–3982, May 2006.
- [70] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *Proceedings of the IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, Kyoto, Japan, November 1-5 2011. IEEE.

-
- [71] C. R. Kube and H. Zhang. Collective robotics: from social insects to robots. *Adapt. Behav.*, 2:189–218, September 1993.
- [72] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.
- [73] E. L. Lawler, J. K. Lenstra, A. R. Kan, and D. B. Shmoys, editors. *The Traveling Salesman Problem: a guided tour of combinatorial optimization*. Wiley, 1985.
- [74] K.-W. Lee, J.-H. Hwang, and D.-S. Kwon. A comparative study between human-human interaction and human-robot interaction. In *New Challenges in Applied Intelligence Technologies*, pages 3–12. Springer, 2008.
- [75] S. G. Loizou and V. Kumar. Mixed initiative control of autonomous vehicles. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pages 1431–1436, Roma, Italy, 10 - 14 April 2007.
- [76] H.-A. Marsiske. IROS: “Roboter waren in Fukushima sehr nützlich”. Heise online, October 2012.
- [77] J. Meyer, P. Schnitzspan, S. Kohlbrecher, K. Petersen, O. Schwahn, M. Andriluka, U. Klingauf, S. Roth, B. Schiele, and O. von Stryk. A semantic world model for urban search and rescue based on heterogeneous sensors. In *RoboCup 2010: Robot Soccer World Cup XIV*, pages 180–193, 2010.
- [78] N. Michael, J. Fink, S. Loizou, and V. Kumar. Architecture, abstractions, and algorithms for controlling large teams of robots: Experimental testbed and results. In *Proc. of the Int. Symposium on Robotics Research*, Hiroshima, Japan, Nov. 2007.
- [79] N. Michael, M. M. Zavlanos, V. Kumar, and G. J. Pappas. Distributed multi-robot task assignment and formation control. In *IEEE International Conference on Robotics and Automation (ICRA) 2008*, pages 128–133, May 2008.
- [80] C. A. Miller, H. B. Funk, M. Dorneich, and S. D. Whitlow. A playbook interface for mixed initiative control of multiple unmanned vehicle teams. In *Proceedings of the 21st Digital Avionics Systems Conference*, volume 2, pages 7E4–1 – 7E4–13, 2002.
- [81] G. A. Mills-Tettey, A. T. Stentz, and M. B. Dias. The dynamic hungarian algorithm for the assignment problem with changing costs. Technical Report CMU-RI-TR-07-27, Robotics Institute, CMU, Pittsburgh, PA, July 2007.
- [82] R. Murphy. Draganflyer credited with first live save with a search and rescue robot!, May 2013.
- [83] R. Murphy, J. Blitch, and J. Casper. AAI/RoboCup-2001 Urban Search and Rescue Events: Reality and Competition. *AI Magazine*, 23(1):37 – 42, 2002.
- [84] R. R. Murphy. Human-robot interaction in rescue robotics. *IEEE Transactions on Systems, Man, And Cybernetics – Part C: Applications and Reviews*, 34(2):138 – 153, May 2004.

-
- [85] Y. Nevatia, T. Stoyanov, R. Rathnam, M. Pfingsthorn, S. Markov, R. Ambrus, and A. Birk. Augmented autonomy: Improving human-robot team performance in urban search and rescue. In *Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2103 – 2108, Nice, France, Sept, 22-26 2008.
- [86] C. W. Nielsen, M. A. Goodrich, and R. W. Ricks. Ecological interfaces for improving mobile robot teleoperation. *IEEE Transactions on Robotics and Automation*, 23(5):927–941, October 2007.
- [87] D. Norman. How might humans interact with robots. Keynote Address to the DARPA/NSF Workshop on Human-Robot Interaction, 2001.
- [88] D. A. Norman. The ‘problem’ of automation: Inappropriate feedback and interaction, not ‘over-automation’. *Philosophical Transactions of the Royal Society of London, B*, 327:585–593, 1990.
- [89] D. A. Norman. *Turn Signals Are The Facial Expressions Of Automobiles*. Basic Books, May 1993.
- [90] D. R. Olsen and M. A. Goodrich. Metrics for evaluating human-robot interactions. In *PerMIS*, 2003.
- [91] R. Parasuraman, M. Barnes, and K. Cosenzo. Adaptive automation for human-robot teaming in future command and control systems. *The International C2 Journal*, 1(2):43 – 68, 2007. Special Issue on Decision Support for Network-Centric Command and Control.
- [92] R. Parasuraman, R. Molloy, and I. L. Singh. Performance consequences of automation-induced “complacency”. *The International Journal of Aviation Psychology*, 3(1):1–23, 1993.
- [93] R. Parasuraman and V. Riley. Humans and automation: Use, misuse, disuse, abuse. *Human Factors*, 39(2):230 – 253, 1997.
- [94] R. Parasuraman, T. B. Sheridan, and C. D. Wickens. A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics — Part A: Systems and Humans*, 30(3):286–297, May 2000.
- [95] L. E. Parker. Alliance: An architecture for fault tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, April 1998.
- [96] L. E. Parker. Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous Robots*, 12(3):231–255, 2002.
- [97] L. E. Parker. Distributed intelligence: Overview of the field and its application in multi-robot systems. *Journal of Physical Agents*, 2(1):5–14, March 2008.
- [98] L. E. Parker and F. Tang. Building multirobot coalitions through automated task solution synthesis. *Proceedings of the IEEE*, 94(7):1289–1305, July 2006.
- [99] K. Petersen, A. Kleiner, and O. von Stryk. Fast task-sequence allocation for heterogeneous robot teams with a human in the loop. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1648 – 1655, 2013.

-
- [100] K. Petersen, G. Stoll, and O. von Stryk. A supporter behavior for soccer playing humanoid robots. In *RoboCup 2010: Robot Soccer World Cup XIV*, pages 386–396, 2010.
- [101] K. Petersen and O. von Stryk. An event-based communication concept for human supervision of autonomous robot teams. *International Journal on Advances in Intelligent Systems*, 4(3&4):357 – 369, 2011.
- [102] K. Petersen and O. von Stryk. Towards a general communication concept for human supervision of autonomous robot teams. In *Proceedings of the Fourth International Conference on Advances in Computer-Human Interactions (ACHI)*, pages 228 – 235, 2011.
- [103] K. Petersen and O. von Stryk. Application independent supervised autonomy. In *Proceedings of the 10th IEEE International Symposium on Safety Security and Rescue Robotics (SSRR 2012)*, 2012.
- [104] S. Petters, D. Thomas, and O. von Stryk. RoboFrame - a modular software framework for lightweight autonomous robots. In *Proceedings of the Workshop on Measures and Procedures for the Evaluation of Robot Architectures and Middleware of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, USA, Oct. 29 2007.
- [105] D. Pinelle and C. Gutwin. A groupware design framework for loosely coupled workgroups. In *Proceedings of the Ninth European Conference on Computer-Supported Cooperative Work*, pages 65 – 82, Paris, France, 18-22 September 2005.
- [106] W. Prinz. Nessie: An awareness environment for cooperative settings. In *Proceedings of The Sixth European Conference on Computer Supported Cooperative Work*, pages 391 – 410, Copenhagen, Denmark, 12-16 September 1999.
- [107] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. Ros: an open-source robot operating system. In *Proceedings of the 2009 ICRA workshop on Open Source Software*, 2009.
- [108] K. Radkhah, S. Kurowski, and O. von Stryk. Design considerations for a biologically inspired compliant four-legged robot. In *Proceedings of the 2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 598–603, Guilin, Guangxi, China, December 19-23 2009.
- [109] C. Reinl and O. von Stryk. Optimal control of multi-vehicle systems under communication constraints using mixed-integer linear programming. In *Proceedings of the First International Conference on Robot Communication and Coordination (RoboComm)*, Athens, Greece, Oct. 15-17 2007.
- [110] J. M. Riley and M. R. Endsley. The hunt for situation awareness: Human-robot interaction in search and rescue. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, pages 693–697, 2004.
- [111] J. M. Riley and M. R. Endsley. Situation awareness in hri with collaborating remotely piloted vehicles. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, pages 407 – 411, 2005.

-
- [112] S. Sariel, T. Balch, and N. Erdogan. Incremental multi-robot task selection for resource constrained and interrelated tasks. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, USA, Oct 29 - Nov 2 2007.
- [113] P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe. Token approach for role allocation in extreme teams: Analysis and experimental evaluation. In *Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, WETICE '04, pages 397–402, Washington, DC, USA, 2004. IEEE Computer Society.
- [114] P. Scerri, P. Velagapudi, K. Sycara, H. Wang, S.-Y. J. Chien, and M. Lewis. Towards an understanding of the impact of autonomous path planning on victim search in usar. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 383 – 388, Taipei, Taiwan, October 18-22 2010.
- [115] J. Scholtz. Theory and evaluation of human robot interactions. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, 2003.
- [116] B. P. Sellner, F. Heger, L. Hiatt, R. Simmons, and S. Singh. Coordinated multi-agent teams and sliding autonomy for large-scale assembly. *Proceedings of the IEEE - Special Issue on Multi-Robot Systems*, 94(7):1425 – 1444, July 2006.
- [117] R. Sheh, T. Kimura, E. Mihankhah, J. Pellenz, S. Schwertfeger, and J. Suthakorn. The robocuprescue robot league: Guiding robots towards fieldable capabilities. In *2011 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, pages 31–34, 2011.
- [118] T. B. Sheridan. *Supervisory Control*, chapter 38, pages 1025–1052. John Wiley & Sons, Inc., 2006.
- [119] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, Dec. 1980.
- [120] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.
- [121] R. M. Taylor. Situational awareness rating technique (sart): The development of a tool for aircrew systems design. In *Situational Awareness in Aerospace Operations (AGARD-CP-478)*, pages 3/1 – 3/17, Neuilly sur Seine, France, 1990. NATO - AGARD.
- [122] B. Trouvain and C. M. Schlick. A comparative study of multimodal displays for multi-robot supervisory control. In *Proceedings of the 7th international conference on Engineering psychology and cognitive ergonomics*, EPCE'07, pages 184–193, Berlin, Heidelberg, 2007. Springer-Verlag.
- [123] A. Valero, M. Mecella, F. Matia, and D. Nardi. Adaptative human-robot interaction for mobile robots. In *The 17th IEEE International Symposium on Robot and Human Interactive Communication, 2008. RO-MAN 2008*, pages 243–248, 2008.
- [124] I. Viguria, Antidioand Maza and A. Ollero. Set: An algorithm for distributed multirobot task allocation with dynamic negotiation based on task subsets. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3339–3344, Roma, Italy, 10 - 14 April 2007.

-
- [125] C. Wampler. *Concise International Encyclopedia of Robotics: Applications and Automation*, chapter Teleoperators, Supervisory Control. John Wiley and Sons, Inc., 1990.
- [126] J. Wang and M. Lewis. Human control for cooperating robot teams. In *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 9 – 16. ACM, 2007.
- [127] R. Wegner and J. Anderson. Balancing robotic teleoperation and autonomy for urban search and rescue environments. In *Advances in Artificial Intelligence, 17th Conference of the Canadian Society for Computational Studies of Intelligence*, Lecture Notes in Computer Science, pages 16–30. Springer, 2004.
- [128] H. A. Yanco and J. Drury. Classifying human-robot interaction: An updated taxonomy. In *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, number 3, The Hague, The Netherlands, October 2004.
- [129] K. Zhang, E. G. Collins, and A. Barbu. A novel stochastic clustering auction for task allocation in multi-robot teams. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3300 –3307, October 2010.

Own Publications

Journal Papers

Karen Petersen and Oskar von Stryk. An event-based communication concept for human supervision of autonomous robot teams. *International Journal on Advances in Intelligent Systems*, 4(3 & 4):357 – 369, 2011.

Martin Friedmann, Karen Petersen, and Oskar von Stryk. Adequate motion simulation and collision detection for soccer playing humanoid robots. *Robotics and Autonomous Systems*, 57:786–795, 2009.

Conference Papers

Karen Petersen, Alexander Kleiner, and Oskar von Stryk. Fast task-sequence allocation for heterogeneous robot teams with a human in the loop. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2013*, pages 1648–1655, 2013.

Stefan Kohlbrecher, Johannes Meyer, Thorsten Graber, Karen Petersen, Oskar von Stryk, and Uwe Klingauf. Hector open source modules for autonomous mapping and navigation with rescue robots. In *RoboCup 2013: Robot Soccer World Cup XVII*, 2013.

Karen Petersen and Oskar von Stryk. Application independent supervised autonomy. In *Proceedings of the 10th IEEE International Symposium on Safety Security and Rescue Robotics (SSRR 2012)*, 2012.

Stefan Kohlbrecher, Karen Petersen, Gerald Steinbauer, Johannes Maurer, Peter Lepej, Suzana Uran, Rodrigo Ventura, Christian Dornhege, Andreas Hertle, Raymond Sheh, and Johannes Pellenz. Community-driven development of standard software modules for search and rescue robots. In *Proceedings of the 10th IEEE International Symposium on Safety Security and Rescue Robotics (SSRR 2012)*, 2012.

Karen Petersen and Oskar von Stryk. Towards a general communication concept for human supervision of autonomous robot teams. In *Proceedings of the Fourth International Conference on Advances in Computer-Human Interactions (ACHI)*, pages 228 – 235, 2011. Best Paper Award.

Mykhaylo Andriluka, Paul Schnitzspan, Johannes Meyer, Stefan Kohlbrecher, Karen Petersen, Oskar von Stryk, Stefan Roth, and Bernt Schiele. Vision based victim detection from unmanned aerial vehicles. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1740 – 1747, October 18 - 22 2010.

Martin Friedmann, Karen Petersen, and Oskar von Stryk. Advanced simulation of distance sensors for mobile robots. In *Simulation, Modeling, and Programming for Autonomous Robots (SIMPAN 2010)*, volume 6472 of *Lecture Notes in Artificial Intelligence*, pages 63–74. Springer, 2010.

Vinay Sachidananda, Diego Costantini, Christian Reinl, Dominik Haumann, Karen Petersen, Parag S. Mogre, and Abdelmajid Khelil. Simulation and evaluation of mixed-mode environments: Towards higher quality of simulations. In *Simulation, Modeling, and Programming for Autonomous Robots (SIMPAN 2010)*, pages 133 – 143. Springer, 2010.

Johannes Meyer, Paul Schnitzspan, Stefan Kohlbrecher, Karen Petersen, Oliver Schwahn, Mykhaylo Andriluka, Uwe Klingauf, Stefan Roth, Bernt Schiele, and Oskar von Stryk. A semantic world model for urban search and rescue based on heterogeneous sensors. In *RoboCup 2010: Robot Soccer World Cup XIV*, pages 180–193, 2010.

Karen Petersen, Georg Stoll, and Oskar von Stryk. A supporter behavior for soccer playing humanoid robots. In *RoboCup 2010: Robot Soccer World Cup XIV*, pages 386–396, 2010.

Martin Friedmann, Karen Petersen, and Oskar von Stryk. Simulation of multi-robot teams with flexible level of detail. In S. Carpin et al., editor, *Simulation, Modeling and Programming for Autonomous Robots (SIMPAN 2008)*, number 5325 in Lecture Notes in Artificial Intelligence, pages 29–40, Venice, Italy, November 2008. Springer. Best Paper Award.

Matthias Kropff, Christian Reinl, Kim Listmann, Karen Petersen, Katayon Radkhah, Faisal Karim Shaikh, Arthur Herzog, Armin Strobel, Daniel Jacobi, and Oskar von Stryk. Mmulator: Towards a common evaluation platform for mixed mode environments. In S. Carpin et al., editor, *Simulation, Modeling, and Programming for Autonomous Robots (SIMPAN 2008)*, number 5325 in Lecture Notes in Artificial Intelligence, pages 41–52. Springer, November 2008.

Martin Friedmann, Karen Petersen, and Oskar von Stryk. Tailored real-time simulation for teams of humanoid robots. In U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, editors, *RoboCup 2007: Robot Soccer World Cup XI*, volume 5001 of *Lecture Notes in Computer Science/Artificial Intelligence*, pages 425–432, Berlin/Heidelberg, 2008. Springer-Verlag.

Workshop Papers

Martin Friedmann, Karen Petersen, and Oskar von Stryk. Adequate motion simulation and collision detection for soccer playing humanoid robots. In *Proceedings of the 2nd Workshop on Humanoid Soccer Robots at the 2007 IEEE-RAS International Conference on Humanoid Robots*, Pittsburgh, PA, USA, Nov. 29 - Dec. 1 2007.

Wissenschaftlicher Werdegang¹

05/2002	Allgemeine Hochschulreife
10/2002 – 08/2007	Studium der Mathematik an der Technischen Universität Darmstadt
08/2007	Diplom in Mathematik mit Schwerpunkt Informatik
seit 09/2007	Doktorandin am Fachbereich Informatik, Technische Universität Darmstadt
09/2007 – 03/2011	Stipendiatin im Graduiertenkolleg 1362 "Cooperative, Adaptive and Responsive Monitoring in Mixed Mode Environments", Fachbereich Informatik, Technische Universität Darmstadt
seit 04/2011	Wissenschaftliche Mitarbeiterin, Fachbereich Informatik, Technische Universität Darmstadt

Erklärung²

Hiermit erkläre ich, dass ich die vorliegende Arbeit, mit Ausnahme der ausdrücklich genannten Hilfsmittel, selbständig verfasst habe.

¹ gemäß § 20 Abs. 3 der Promotionsordnung der TU Darmstadt

² gemäß § 9 Abs. 1 der Promotionsordnung der TU Darmstadt